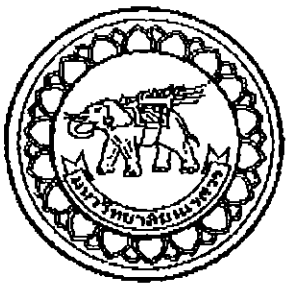


# การบีบอัดข้อมูลภาพโดยวิธีฮัฟฟ์แมน

# IMAGE COMPRESSION BY USING HUFFMAN CODING


นาย มนัสชัย โรจนวิเชียร รหัส 44370336

นางสาวอรอุมา อักโข รหัส 44370567

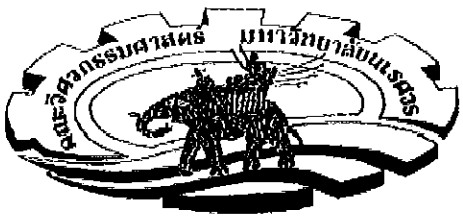



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2547



# ใบรับรองโครงงานวิศวกรรม

| | |
|---|---|
| **หัวข้อโครงงาน** | **การบีบอัดข้อมูลภาพโดยวิธีการฮัฟฟ์แมน** |
| **ผู้เสนอโครงงาน** | นายมนัสชัย โรจนวิเชียร รหัส 44370336 |
| | นางสาวอรอุมา อักโข รหัส 44370567 |
| **อาจารย์ที่ปรึกษา** | ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มเม่น |
| **สาขาวิชา** | วิศวกรรมคอมพิวเตอร์ |
| **ภาควิชา** | วิศวกรรมไฟฟ้าและคอมพิวเตอร์ |
| **ปีการศึกษา** | 2547 |

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มเม่น)

.....กรรมการ
(ดร.พนมขวัญ ริยะมงคล)

.....กรรมการ
(นายแสงชัย มังกรทอง)

| | | | |
|---|---|---|---|
| **หัวข้อโครงงาน** | **การบีบอัดข้อมูลภาพโดยวิธีฮับแมน** | | |
| **ผู้ดำเนินโครงการ** | **นายมนัสชัย โรจนวิเชียร** | **รหัส** | 44370336 |
| | **นางสาวอรอุมา อักโข** | **รหัส** | 44370567 |
| **อาจารย์ที่ปรึกษา** | **ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มแม่น** | | |
| **สาขาวิชา** | **วิศวกรรมคอมพิวเตอร์** | | |
| **ภาควิชา** | **วิศวกรรมไฟฟ้าและคอมพิวเตอร์** | | |
| **ปีการศึกษา** | 2547 | | |

.....

## บทคัดย่อ

โครงงานนี้เป็นการศึกษาและพัฒนาด้วยโปรแกรมแมทแล็ปสำหรับการบีบอัดข้อมูล 1 มิติ หรือข้อมูล 2 มิติ โดยใช้วิธีการแปลรหัสแบบฮัฟฟ์แมน ผลที่ได้จากโครงงานนี้ จะได้ค่าอัตราการบีบอัดสูงสุดสำหรับข้อมูล 1 มิติ คือ 1.30 และ สำหรับข้อมูล 2 มิติ คือ 1.32 นอกจากนี้ค่า SNR และ PSNR มีค่าเข้าใกล้เลขอนันต์ทั้งสองกรณี นั้นคือการคลายการบีบอัดข้อมูลสมบูรณ์แบบเมื่อนำมาเปรียบเทียบกับข้อมูลต้นแบบของทั้งสองกรณี

| | |
|---|---|
| **Project Title** | IMAGE COMPRESSION BY USING HUFFMAN CODING |
| **Name** | Mr. Manutchai Rojanavichain ID. 44370336 |
| | Miss Onuma Akkho ID. 44370567 |
| **Project Advisor** | Assistant Professor Suchart Yammen, Ph.D. |
| **Major** | Computer Engineering |
| **Department** | Electrical and Computer Engineering |
| **Academic** | 2004 |

.....

## ABSTRACT

This project is to study and develop a Mathlab program for compressing either one-dimensional or two-dimensional data by using the Huffman Coding method. Form the project results, the maximum compression ratio is 1.30 for the one-dimensional data, and is 1.32 for the two-dimensional data. Furthermore, both the SNR and PSNR values approach infinity; that is, the decompressed data is perfectly restored when compared with the original data in the two cases.

# กิตติกรรมประกาศ

โครงงานวิศวกรรมคอมพิวเตอร์สำเร็จได้ด้วยดี ผู้จัดทำได้รับความกรุณาอย่างยิ่งจาก ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มเม่น อาจารย์ที่ปรึกษาโครงงาน ที่ได้กรุณาให้คำปรึกษา แนะนำ และข้อคิดเห็นในการแก้ปัญหาต่างๆ ตลอดจนการตรวจแก้ไขหนังสือประกอบโครงงานนี้จนสำเร็จสมบูรณ์

ท้ายสุดนี้ คณะผู้จัดทำขอขอบคุณท่านคณะกรรมการสอบโครงงานทุกท่านเป็นอย่างสูงที่ได้ช่วยพิจารณา ให้คำแนะนำการตรวจทานแก้ไข และอนุมัติให้โครงงานนี้สำเร็จลุล่วงได้ด้วยดี

นายมนัสชัย โรจนวิเชียร
นางสาวอรอุมา อักโข

# สารบัญ

หน้า

บทคัดย่อภาษาไทย.....ก
บทคัดย่อภาษาอังกฤษ.....ข
กิตติกรรมประกาศ.....ค
สารบัญ.....ง
สารบัญตาราง.....ฉ
สารบัญรูป.....ช

บทที่ 1 บทนำ
1.1 ความเป็นมาและความสำคัญของปัญหา.....1
1.2 วัตถุประสงค์ของโครงงาน.....1
1.3 ขอบข่ายของโครงงาน.....2
1.4 ขั้นตอนการดำเนินโครงงาน.....2
1.5 ผลที่คาดว่าจะได้รับ.....2
1.6 รายละเอียดงบประมาณตลอดโครงงาน.....3

บทที่ 2 หลักการและทฤษฎี
2.1 หลักการบีบอัดข้อมูลภาพ.....4
2.2 การเข้ารหัส.....4
2.3 อัลกอริทึมของการเข้ารหัสแบบฮัฟฟ์แมน.....4
2.4 กระบวนการของการเข้ารหัสแบบฮัฟฟ์แมน.....5
2.5 อัตราส่วนของการบีบอัดข้อมูล.....10
2.6 การหาความผิดพลาดของการบีบอัดข้อมูล.....10

บทที่ 3 ขั้นตอนการโครงงาน
3.1 ศึกษาอัลกอริทึมของรหัสฮัฟฟ์แมน.....12
3.2 โปรแกรมการสร้างตารางรหัสฮัฟฟ์แมน.....13
3.3 โปรแกรมเข้ารหัสฮัฟฟ์แมน.....15
3.3.1 ฟังก์ชันการเข้ารหัสฮัฟฟ์แมน.....15

# สารบัญ (ต่อ)

หน้า

3.4 โปรแกรมการถอดรหัสฮัฟฟ์แมน....17
3.4.1 ฟังก์ชันการถอดรหัสฮัฟฟ์แมน....17

บทที่ 4 การทดลอง

4.1 วัตถุประสงค์การทดลอง....20
4.2 การเรียกใช้โปรแกรม....20
4.3 การทดลองเข้ารหัสและการถอดรหัสฮัฟฟ์แมน....21
4.3.1 ผลการทดลองการบีบอัดข้อมูลภาพ....21
4.3.2 ผลการทดลองการบีบอัดข้อมูลอักษร....25
4.4 ผลการวิเคราะห์ข้อมูล....27

บทที่ 5 บทสรุป

5.1 สรุปผลการทดลอง....28
5.2 ปัญหาในการทดลองและแนวทางแก้ไข....28
5.3 แนวทางการพัฒนาในอนาคต....28

เอกสารอ้างอิง

ภาคผนวก

ประวัติผู้เขียนโครงงาน

# สารบัญตาราง

ตารางที่ หน้า

1.1 ตารางปฏิบัติงาน....2
2.1 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 1....6
2.2 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 2....6
2.3 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 3....6
2.4 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 4....7
2.5 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 5....7
2.6 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 6....8
4.1 สรุปผลที่ได้จากการบีบอัดข้อมูลภาพโดยวิธีฮัฟฟ์แมนทั้ง 4 กรณี....25
4.2 สรุปผลที่ได้จากการบีบอัดข้อมูลอักษรโดยวิธีฮัฟฟ์แมนทั้ง 2 กรณี....26

# สารบัญรูป

รูปที่ หน้า

2.1 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 1....9

2.2 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 2....9

2.3 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 3....9

2.4 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 4....10

3.1 ขั้นตอนการทำโครงงาน....12

3.2 Flowchart โปรแกรมการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมน....13

3.3 Flowchart ฟังก์ชันสร้างตารางฮัฟฟ์แมน....14

3.4 แผนภาพโปรแกรมการเข้ารหัสฮัฟฟ์แมน (Huffman Encode)....15

3.5 Flowchart ฟังก์ชันการเข้ารหัสแบบฮัฟฟ์แมน....16

3.6 แผนภาพโปรแกรมการถอดรหัสฮัฟฟ์แมน (Huffman Decode)....17

3.7 Flowchart ฟังก์ชันการถอดรหัสฮัฟฟ์แมน....18

4.1 Flowchart โปรแกรมการบีบอัดข้อมูลอักษรแบบฮัฟฟ์แมน....20

4.2 แสดงภาพ Lena ก่อนการบีบอัดและหลังการบีบอัดข้อมูลภาพด้วยฮัฟฟ์แมน....21

4.3 แสดงภาพ Nemo ก่อนการบีบอัดและหลังการบีบอัดข้อมูลภาพด้วยฮัฟฟ์แมน....22

4.4 แสดงภาพ Baby ก่อนการบีบอัดและหลังการบีบอัดข้อมูลภาพด้วยฮัฟฟ์แมน....23

4.5 แสดงภาพ Eye ก่อนการบีบอัดและหลังการบีบอัดข้อมูลภาพด้วยฮัฟฟ์แมน....24

4.6 แสดงผลการทดลองไฟล์ go.txt....25

4.7 แสดงผลการทดลองไฟล์ name.txt....26

# บทที่ 1
# บทนำ

## 1.1 ความเป็นมาและความสำคัญของปัญหา

ยุคปัจจุบัน เป็นยุคของข่าวสารข้อมูลไม่ว่าจะเป็นข้อความหรือรูปภาพ จำเป็นจะต้องใช้เนื้อที่ในการจัดเก็บเป็นจำนวนมาก โดยเฉพาะอย่างยิ่งข้อมูลรูปภาพ ตัวอย่างเช่นหน้าเอกสารขนาด A4 ที่ทำการสแกนที่ระดับความละเอียด 300 จุดต่อนิ้ว สำหรับเอกสารภาพสี จะมีขนาดของแฟ้มภาพเอกสารที่ประมาน 25 MB ถึงแม้ว่าเทคโนโลยีทางด้านอุปกรณ์เก็บข้อมูลจะได้รับการพัฒนาให้มีประสิทธิภาพเพียงใดก็ตาม แต่ก็ยังประสบปัญหาอุปกรณ์เก็บข้อมูลมีไม่เพียงพอ เนื่องจากข้อมูลมีแนวโน้มเพิ่มมากขึ้นตลอดเวลา

การแก้ปัญหาดังกล่าวข้างต้นสามารถทำได้โดยการบีบข้อมูลเพื่อลดขนาดข้อมูลในเนื้อที่จัดเก็บให้มีขนาดเล็กลง ทำให้อุปกรณ์สามารถเก็บข้อมูลได้เพิ่มมากขึ้น นอกจากนี้ขนาดข้อมูลที่เล็กลงยังเป็นการช่วยประหยัดเวลาในการสื่อสารข้อมูลได้วิธีหนึ่ง

การบีบข้อมูลภาพที่ดีจะต้องให้อัตราส่วนการบีบอัด(Compression ratio) สูงและคุณภาพของภาพต้องคมชัดและสมบูรณ์ หนึ่งในมาตรฐานการบีบอัดภาพที่ใช้กันอย่างแพร่หลายในปัจจุบัน คือ เทคนิคการบีบอัดภาพแบบเจเพ็ก (JPEG หรือ Joint Photographic Expert Group) ซึ่งมีความสามารถบีบอัดภาพแบบสูญเสียข้อมูลบางส่วนได้ในอัตราส่วนการบีบ 20:1 ถึง 30:1

แต่เมื่ออัตราการบีบสูงขึ้น คุณภาพของภาพจะลดลง การเข้ารหัสแบบฮัฟแมน(Huffman coding) เป็นอีกทางเลือกหนึ่งที่สามารถนำมาใช้ในการบีบอัดไฟล์ภาพ โดยไม่ทำให้ภาพที่ผ่านการบีบอัด มีคุณภาพที่เปลี่ยนไป จะได้ภาพที่มีขนาดเล็กลงแต่มีคุณภาพใกล้เคียงกับภาพเดิม ปัจจุบันมีการนำหลักการของฮับแมนมาพัฒนาขึ้นเป็นโปรแกรมในภาษาต่างๆ และในการใช้โปรแกรม MATLAB ยังมีน้อยมาก ซึ่งหากมีการนำมาพัฒนาก็จะนำมาใช้เอื้อประโยชน์ในด้านดังกล่าว

## 1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาเทคนิคการเข้ารหัสแบบฮัฟฟ์แมน
2. เพื่อลดจำนวนบิตของข้อมูลภาพโดยไม่ทำให้ข้อมูลภาพเปลี่ยนแปลง
3. เพื่อออกแบบโปรแกรม MATLAB โดยใช้อัลกอริทึ่มของฮัฟฟ์แมน

## 1.3 ขอบข่ายของโครงงาน

1. ศึกษาข้อมูลการเข้ารหัสแบบฮัฟฟ์แมน
2. ศึกษาการเขียนโปรแกรมโดยใช้ MATLAB

## 1.4 ขั้นตอนการดำเนินโครงงาน

ตารางที่ 1.1 ตารางปฏิบัติงาน

| กิจกรรม | พ.ย. 46 | ธ.ค. 46 | ม.ค. 47 | ก.พ 47 | มี.ค. 47 | เม.ย 47 | พ.ค 47 | ม.ค. 48 | ก.พ 48 | มี.ค 48 | เม.ย 48 | พ.ค 48 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ค้นคว้าข้อมูลการเข้ารหัสแบบฮับแมน | ← | ― | → | | | | | | | | | |
| ศึกษาอัลกอรึทึมของฮับแมน | | | | ← | ― | → | | | | | | |
| ออกแบบโปรแกรม MATLAB | | | | | | ← | ― | ― | → | | | |
| ทดสอบโปรแกรมและทำการแก้ไขจุดบกพร่อง | | | | | | | | ← | ― | ― | → | |
| นำเสนอผลงานและจัดทำรายงาน | | | | | | | | | | | ← | → |

## 1.5 ผลที่คาดว่าจะได้รับ

1. โปรแกรมการเข้ารหัสแบบฮัฟฟ์แมนโดยใช้ MATLAB
2. เพื่อนำไปใช้ในการบีบอัดข้อมูลภาพให้มีขนาดเล็กลง
3. เพื่อให้ผู้สนใจศึกษาการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนได้นำโปรแกรมไปพัฒนา หรือนำไปใช้ต่อไป

**1.6 รายละเอียดงบประมาณตลอดโครงงาน**

| | | | |
|---|---|---|---|
| 1. ค่าวัสดุสำนักงาน | เป็นจำนวน | 500 | บาท |
| 2. ค่าจ้างถ่ายเอกสาร | เป็นจำนวน | 1000 | บาท |
| 3. ค่าจ้างเข้าเล่ม | เป็นจำนวน | 500 | บาท |
| | รวมเป็นเงินทั้งสิ้น | 2,000 | บาท |

**หมายเหตุ** ขออนุมัติถัวเฉลี่ยทุกรายการ

# บทที่ 2
# หลักการและทฤษฎี

## 2.1 หลักการบีบอัดข้อมูลภาพ (Image Compression)

ความสนใจในการบีบอัดข้อมูลภาพเริ่มขึ้นเมื่อประมาณกว่า 35 ปีที่แล้วในโลกของอะนาลอก เพื่อลดแบนด์วิธในการส่งสัญญาณวีดีโอ โดยกระบวนการนี้ถูกเรียกว่า การบีบแบนด์วิธ (Bandwidth compression) จากนั้นเมื่อเทคโนโลยีด้านดิจิทัลพัฒนาขึ้นทำให้เกิดการบีบข้อมูลทางดิจิทัลขึ้น และได้รับการพัฒนาเพิ่มขึ้นอย่างรวดเร็ว

หลักในการบีบอัดข้อมูลภาพคือ พยายามลดหรือกำจัดส่วนของข้อมูลที่เกินความจำเป็นหรือซ้ำซ้อนกัน (Data redundancy) โดยยังคงข่าวสารไว้เหมือนเดิม ซึ่งจะทำให้ข้อมูลภาพมีขนาดลดลงจากเดิม และสามารถนำภาพกลับมาแสดงภายหลังโดยผ่านกระบวนการคลาย (Decompression)

**ประเภทของการบีบอัดข้อมูลภาพแบ่งได้เป็น 2 แบบคือ**

2.1.1 การบีบอัดข้อมูลแบบไม่มีการสูญเสีย (Lossless compression หรือ Bit-preserving หรือ Versible compression)

2.1.2 การบีบอัดข้อมูลแบบมีการส่วนเสียบางส่วน (Lossy Compression หรือ Irreversible compression)

## 2.2 การเข้ารหัส (coding)

กระบวนการที่กำหนดให้การจัดลำดับของเลขฐานสอง ไปเป็นสัญลักษณ์ เรียกว่า การเข้ารหัส (coding) เซตของการจัดลำดับของเลขฐานสอง เรียกว่า รหัส และเลขที่เป็นเลขเฉพาะของเซต เรียก รหัสคำ(codeword)

การเข้ารหัสที่เราสนใจ คือ การเข้ารหัสเอนโทรปี(Entropy coding) คือการลดความฟุ่มเฟือยด้วยวิธีการบีบอัดข้อมูลโดยไม่สนใจถึงความหมายของข้อมูล เพียงแต่มองข้อมูลเป็นบิต 0 หรือ 1 ซึ่งในการบีบอัดภาพที่นิยมอยู่ คือ การเข้ารหัสแบบฮัฟฟ์แมน (Huffman coding)

## 2.3 อัลกอริทึมของ การเข้ารหัสแบบฮัฟฟ์แมน (Huffman coding Algorithm)

กระบวนการนี้ถูกพัฒนาขึ้นโดย เดวิท ฮัฟฟ์แมน(David Huffman) ซึ่งเป็นสมาชิกในทฤษฎีข่าวสารแรกของ โรเบิร์ต ฟาโน(Robert Fano) ที่ MIT การเกิดรหัสโดยการใช้กระบวนการนี้ จึงเรียกว่ารหัสของฮัฟฟ์แมน เป็นรหัสแบบ prefix code และเป็นสถานการณ์ที่ให้แบบตัวอย่าง(เซตของความน่าจะเป็น)

**กระบวนการของฮัฟฟ์แมนมีข้อสังเกตุพื้นฐาน 2 ข้อคือ**

2.3.1. ในรหัสข้อมูลปัจจุบัน สัญลักษณ์ที่มีอัตราการเกิดบ่อยครั้ง(มีความน่าจะเป็นในการเกิดสูง) จะมีจำนวนบิตของรหัสข้อมูลสั้นลง

2.3.2. ในรหัสข้อมูลปัจจุบัน สัญลักษณ์สองตัวที่มีการปรากฏน้อยที่สุดจะมีความยาวเท่ากัน

วิธีการสังเกตุ คือ ถ้าสัญลักษณ์มีอัตราการเกิดขึ้นสูง จะมีรหัสคำยาวกว่า จำนวนเฉลี่ยบิตต่อสัญลักษณ์มากกว่า ในเงื่อนไขการเปลี่ยนตำแหน่ง ดังนั้นจึงมีการกำหนดให้ รหัสคำที่ยาวกว่าเป็นสัญลักษณ์ที่มีความถี่ในของการเกิดสูงกว่า ไม่สามารถเป็น รหัสปัจจุบัน

การสังเกตุอีกวิธีหนึ่ง พิจารณาตามตำแหน่ง โดยการคาดคะเน จากรหัสที่เข้ามาที่มีรหัสคำเหมือนกันมีความถี่ในการเกิดต่ำ สัญลักษณ์จะไม่มีความยาวเท่ากัน สมมุติให้ความยาวของ รหัสคำคือ คือ kบิต ยาวกว่ารหัสคำที่สั้น รหัสคำที่สั้นกว่าจะไม่เป็น prefix ของรหัสคำที่ยาวกว่า หมายความว่าถ้าเราตกอยู่ที่ kบิตตัวสุดท้ายของรหัสคำที่ยาวกว่ารหัสคำอีกตัวหนึ่งจะมีความชัดเจนขึ้น

เนื่องจาก รหัสคำที่เหมือนกันที่มีอัตราการเกิดต่ำที่สุด ในสัญลักษณ์ที่เป็นกลุ่มตัวอักขระ ไม่มีรหัสคำตัวใดยาวกว่าตัวนี้ได้ ดังนั้นจึงไม่มีผลกระทบอันใดในการทำให้รหัสคำสั้นลง โดยจะเริ่มที่การเข้ารหัสพรีฟิก ของตัวอื่น เมื่อทำการตกที่ k บิต เราจะได้รหัสใหม่ที่มีความยาวเฉลี่ยสั้นกว่าเดิม

## 2.4 กระบวนการของการเข้ารหัสแบบฮับแมน

จะมีวิธีการโดยรวมค่าอัตราการเกิดของสัญลักษณ์2ตัวที่มีค่าอัตราการเกิดต่ำที่สุด จะมีความแตกต่างที่บิตสุดท้ายเท่านั้น นั่นคือ ถ้า $\gamma$ และ $\delta$ คือสัญลักษณ์ 2ตัวสุดท้ายที่มีอัตราการเกิดต่ำที่สุด ในสัญลักษณ์ของตัวอักขระ และถ้ารหัสคำของ $\gamma$ คือ m*0 รหัสคำของ$\delta$ จะเป็น m*1 โดยที่ m คือ ค่าของ 1s และ 0s และ *

ข้อมูลแต่ละตัว จะถูกแปลงไปเป็น ชุดของ ตัวเลข 0 และ 1 ซึ่งจะเป็นตัวแสดงถึง เส้นทาง ของมันในแผนภูมิรูปต้นไม้ของ การเข้ารหัสข้อมูล แบบฮัฟฟ์แมน (Huffman encoding) โดยข้อมูลแต่ละตัวจะมีความยาวของชุดตัวเลขดังกล่าว ต่างกันไป ขึ้นกับ ค่าความถี่ ของมันที่ปรากฏในข้อมูลทั้งหมด

ข้อมูลที่มี ความถี่สูงสุด จะมีความยาวของชุดตัวเลขน้อยสุดคือ 2 ตัว (2 บิท) ส่วนข้อมูลที่มีความถี่ต่ำสุดจะมีความยาวของชุดตัวเลข 2N-1 ตัว เมื่อ N คือค่าบิทของการจัดเก็บข้อมูลปกติ

ยกตัวอย่างเช่น ข้อมูล 8 บิท มีข้อมูลที่เป็นไปได้ 256 ค่า (0-255) เมื่อเข้ารหัสแบบฮัฟฟ์แมน จะมีความยาวของชุดตัวเลขเป็นได้ ตั้งแต่ 2 ตัว (ความถี่สูงสุด) จนถึง 128 ตัว (ความถี่ต่ำสุด)

ในการ ถอดรหัสข้อมูล จะดูจากเส้นทางของข้อมูล เริ่มจาก ส่วนยอด ลงมาไปเรื่อยจนครบทุกตัวเลขที่เป็น ไปได้สำหรับข้อมูลตัวหนึ่ง ๆ จากนั้นจึงเริ่มดูจากยอดลงมาสำหรับข้อมูลตัวถัดไป เป็นเช่นนี้ไปเรื่อย ๆ

ถึงแม้ การเข้ารหัสแบบฮัฟฟ์แมน มักจะช่วยทำให้ความจุของข้อมูล ลดลง ไม่มากนัก แต่มันจะช่วยรักษาตัวข้อมูลเดิม ไว้ได้ทั้งหมด เมื่อถอดรหัสออกมา

**ตัวอย่างที่ 1 การทำงานการเข้ารหัสแบบฮัฟฟ์แมน โดยการใช้ Table**

**ตารางที่ 2.1**

| Letter | Probability | Codeword |
|---|---|---|
| a1 | 0.2 | c(a1) |
| a2 | 0.4 | c(a2) |
| a3 | 0.2 | c(a3) |
| a4 | 0.1 | c(a4) |
| a5 | 0.1 | c(a5) |

**ขั้นตอนที่1** ทำการเรียงสัญลักษณ์ใหม่โดยเรียงจากอัตราการการเกิด จากมากไปน้อย

ตารางที่ 2.2

| Letter | Probability | Codeword |
|---|---|---|
| a2 | 0.4 | c(a2) |
| a1 | 0.2 | c(a1) |
| a3 | 0.2 | c(a3) |
| a4 | 0.1 | c(a4) |
| a5 | 0.1 | c(a5) |

**ขั้นตอนที่2** สัญลักษณ์ที่มีอัตราการเกิดต่ำที่สุด2ตัวสุดท้าย ในที่นี้คือ a4 และ a5 **นำค่าอัตราการเกิดรวมกันจะได้ค่า ของอัตราการเกิดใหม่ และกำหนดให้ รหัสคำใหม่ คือ $\alpha$ ดังตาราง ที่2.2**

**ตารางที่ 2.3**

| Letter | Probability | Codeword |
|---|---|---|
| a2 | 0.4 | c(a2) |
| a1 | 0.2 | c(a1) |
| a3 | 0.2 | c(a3) |
| a4 | 0.2 | $\alpha$1 |

ทำขั้นตอนที่1และ2ไปเรื่อยๆจนกว่าจะได้ค่าที่เหลือสุดท้าย 2 ค่า ดังตารางที่ 2.4 และตารางที่ 2.5

**ตารางที่ 2.4**

| Letter | Probability | Codeword |
|---|---|---|
| a2 | 0.4 | c(a2) |
| a3 | 0.4 | $\alpha 2$ |
| a1 | 0.2 | c(a1) |

**ตารางที่ 2.5**

| Letter | Probability | Codeword |
|---|---|---|
| a3 | 0.6 | $\alpha 3$ |
| a2 | 0.4 | c(a2) |

**ขั้นตอนที่3** กำหนดค่าโคทเวิร์ดให้กับสองค่าสุดท้ายโดยให้ค่าที่มีอัตราการเกิดสูงสุดมีค่าเท่ากับ **0** และค่าที่มีอัตราการเกิดต่ำกว่ามีค่าเท่ากับ **1**

$$\alpha 3 = 0$$
$$c(a2) = 1$$

จาก$\alpha 3$ คือ ค่าของ $\alpha 2 + c(a1)$ ดังนั้นจะได้

$$\alpha 2 = (\alpha 3 * 0) = 00$$
$$c(a1) = (\alpha 3 * 1) = 01$$

จาก$\alpha 2$ คือ ค่าของ $c(a3) + \alpha 1$ ดังนั้นจะได้

$$c(a3) = (\alpha 2 * 0) = 000$$
$$\alpha 1 = (\alpha 2 * 1) = 001$$

จาก$\alpha 1$ คือ ค่าของ c(a4) +c(a5) ดังนั้นจะได้

$$c(a4) = (\alpha 1 * 0) = 0010$$

$$c(a5) = (\alpha 1 * 1) = 0011$$

เราจะได้ผลของการถอดรหัสแบบฮัฟฟ์แมนได้ดังตารางที่2.6

**ตารางที่ 2.6**

| Letter | Probability | Codeword |
|---|---|---|
| a2 | 0.4 | 1 |
| a1 | 0.2 | 01 |
| a3 | 0.2 | 000 |
| a4 | 0.1 | 0010 |
| a5 | 0.1 | 0011 |

**ตัวอย่างที่ 2** การทำงานการเข้ารหัสแบบฮัฟฟ์แมน โดยใช้แผนภูมิต้นไม้

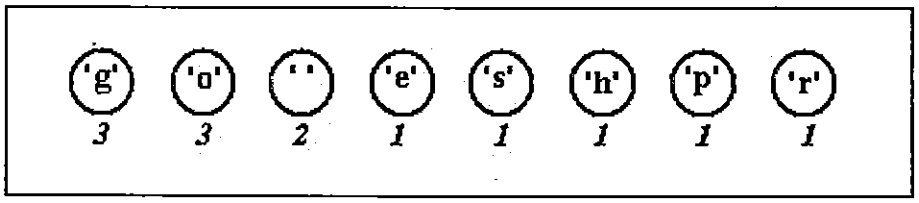
ยกตัวอย่าง ประโยค "GO GO GOPHERS " ซึ่งมีทั้งหมด 13 อักขระ (รวมช่องว่างภายใน) สามารถจัดเก็บได้ในเครื่องขนาด 8 บิท โดยใช้หน่วยความจำทั้งสิ้น 104 บิท (1 อักขระ = 8 บิท) แต่ถ้าเก็บแบบ 3 บิท จะใช้หน่วยความจำทั้งสิ้นเพียง 39 บิท ซึ่งถือเป็นค่า ต่ำสุด ที่เราต้องใช้สำหรับการเก็บข้อมูลชุดนี้ตามปกติ

อย่างไรก็ตาม ประโยคนี้เมื่อผ่านการเข้ารหัส แบบฮัฟฟ์แมน จะทำให้มีความจุเหลือเพียง 37 บิท เท่านั้น ดังมีรายละเอียดดังที่เห็น

สำหรับข้อมูลเพิ่มเติมจากนี้ ดูได้จาก http://www.cs.duke.edu/csed/poop/huff/info ตัวอย่างการเข้ารหัสแบบฮัฟฟ์แมน (Huffman Encoding)

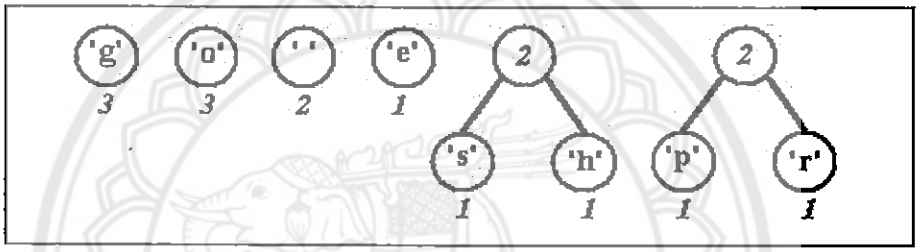
เราสามารถจัดเก็บข้อมูลในประโยค "GO GO GOPHERS " ซึ่งมีทั้งหมด 13 อักขระ (รวมช่องว่างภายใน) โดยการเข้ารหัส แบบฮัฟฟ์แมน ได้ดังนี้

1. คำนวณ ค่าความถี่ ในการปรากฏของแต่ละอักขระในประโยค ได้ผลดังนี้



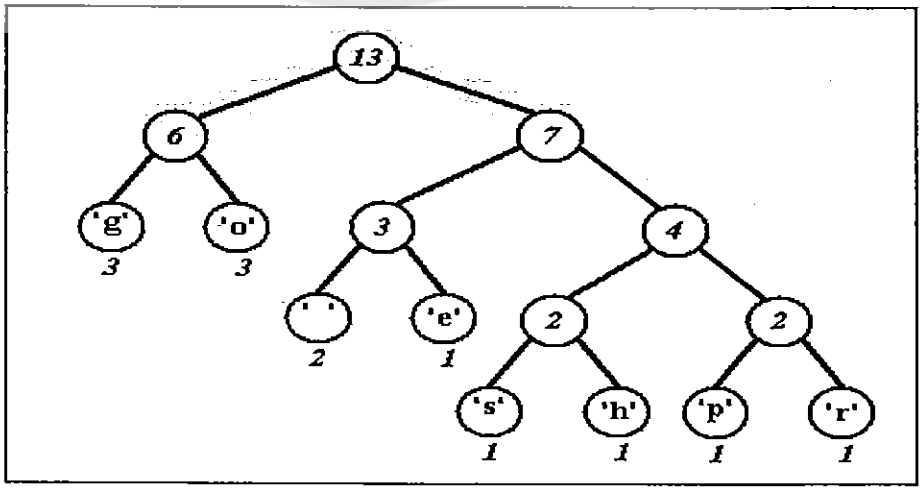

**รูปที่ 2.1**

2. สร้าง แผนภูมิต้นไม้ โดยนำเอาค่าความถี่ น้อยที่สุด ในชุด มาบวกกันครั้งละ 2 ตัว ได้เป็นค่าความถี่ใหม่ขึ้นมา




รูปที่ 2.2

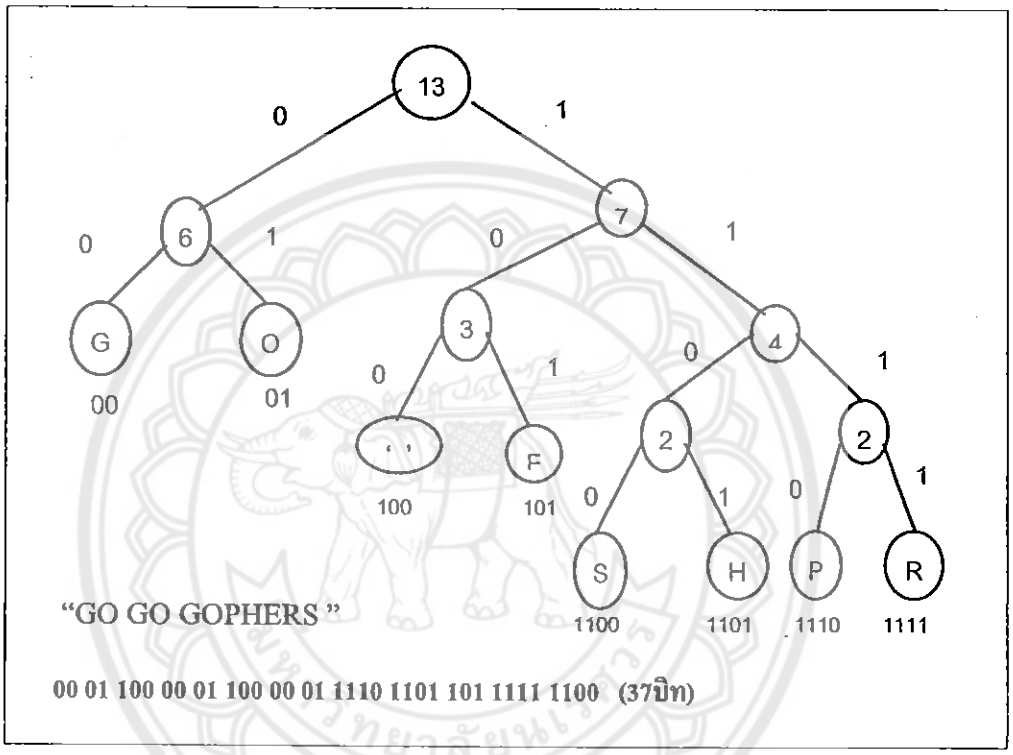
3. จากนั้นดำเนินการแบบเดิม (ตามขั้นที่ 2) ไปเรื่อย ๆ จนกระทั่ง ได้ค่าความถี่รวมสูงสุดค่าเดียว อยู่บนสุด ซึ่งจะมีค่าเท่ากับจำนวนอักขระในชุดข้อมูลดังกล่าวพอดี (คือ 13 ในที่นี้) สุดท้ายเราจะได้ แผนภูมิต้นไม้ ดังแสดงในรูปที่ 3




**รูปที่ 2.3**

จากนั้น จึงมาสร้างเส้นทางของอักขระแต่ละตัว นับจากส่วนยอดลงมา โดยกำหนดให้การเคลื่อนที่ ไปทางซ้ายแทนด้วย 0 และ ทางขวาด้วย 1 ได้ว่า G = 00, O = 01, _ = 100, E = 101, S = 1100, H = 1101, P = 1110,และ R = 1111

ด้วยเหตุนี้เราจึงเขียนประโยค "GO GO GOPHERS " ใหม่โดยใช้การเข้ารหัสแบบฮัฟฟ์แมนข้างต้นได้เป็น 00 01 100 00 01 100 00 01 1110 1101 101 1111 1100 ซึ่งใช้หน่วยความจำเพียง 37 บิท เท่านั้น




รูปที่ 2.4

## 2.5 อัตราส่วนของการบีบอัดข้อมูล (Compression Ratio : CR)

คือ การเปรียบเทียบค่าระหว่างขนาดของข้อมูลต้นฉบับ (n1) กับข้อมูลที่ถูกบีบอัดข้อมูลแล้ว (n2)

$$CR = \frac{n1}{n2} \qquad (1)$$

## 2.6 การหาความผิดพลาดของการบีบอัดข้อมูล

เราสามารถใช้ค่าของ Signal to Noise Ratio (SNR) และ Peak Signal to Noise Ratio (PSNR) โดยจะเป็นตัววัดคุณภาพของการบีบอัดข้อมูลว่า เมื่อบีบอัดข้อมูลแล้วคุณภาพของภาพที่ได้ใหม่จะมีการสูญเสียมากน้อยเพียงใด

SNR ถูกนิยามในหน่วย (dB) ดังต่อไปนี้

$$SNR_{(dB)} = 10\log_{10}\left\{\frac{\left[\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left(s(m,n)\right)^2\right]}{\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left(s(m,n)-\hat{s}(m,n)\right)^2}\right\} \quad (2)$$

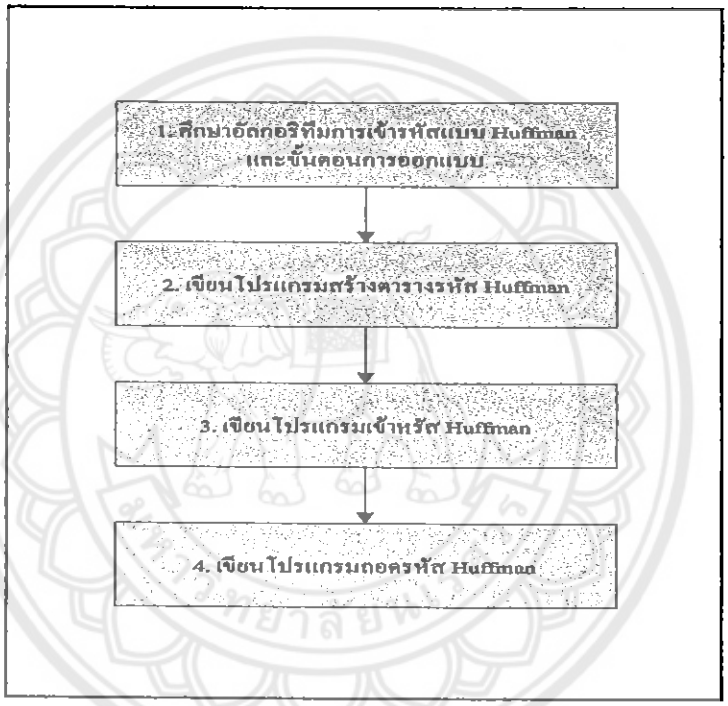
โดยที่ $s(m,n)$ และ $\hat{s}(m,n)$ คือ ค่าข้อมูลภาพต้นฉบับและภาพที่ถูกคลายหลังการบีบอัดแล้วตามลำดับ สำหรับ m และ n เป็นค่าดัชนีตามแนวนอนและแนวตั้งของภาพทั้งสอง หรืออาจวัดค่าความผิดพลาดของการบีบอัดข้อมูลในรูปแบบของ PSNR ดังต่อไปนี้

$$PSNR_{(dB)} = 10\log_{10}\left\{\frac{\max\left|s(m,n)\right|}{\frac{1}{MN}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left(s(m,n)-\hat{s}(m,n)\right)^2}\right\} \quad (3)$$

โดยที่ $\max\left|s(m,n)\right|$ คือ ค่าสูงสุดของข้อมูลภาพต้นแบบและค่า MN คือขนาดของข้อมูลภาพ

# บทที่ 3
# ขั้นตอนการดำเนินโครงงาน

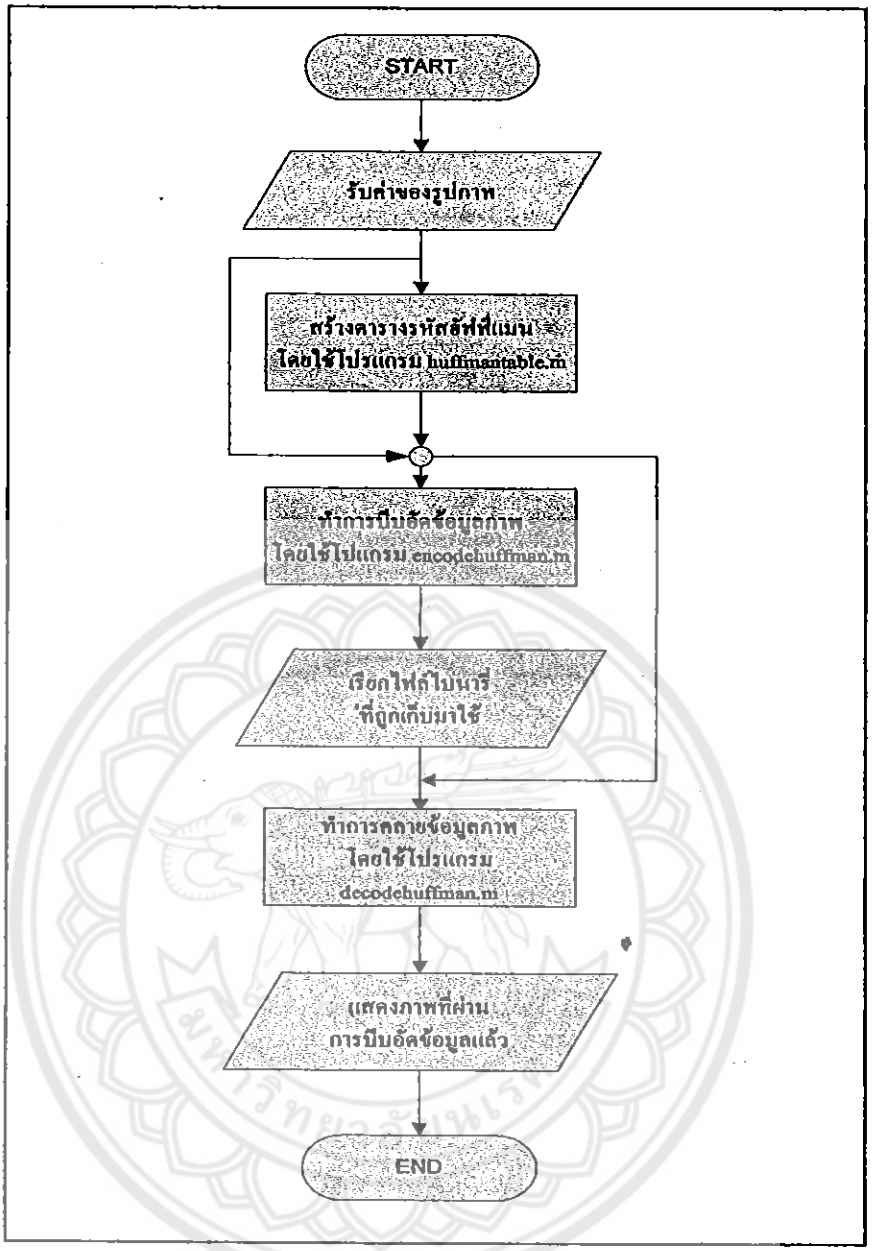
ในการบีบอัดข้อมูลภาพโดยวิธีการฮัฟฟ์แมน ได้แบ่งขั้นตอนการโครงงานออกเป็น 4 ขั้นตอน โดยเริ่มจากการศึกษาอัลกอริทึมการเข้ารหัสแบบ Huffman และขั้นตอนการออกแบบ เขียนโปรแกรมสร้างตารางรหัสฮัฟฟ์แมน เขียนโปรแกรมเข้ารหัสฮัฟฟ์แมน และเขียนโปรแกรมถอดรหัสฮัฟฟ์แมน แสดงดังรูปที่ 3.1




รูปที่ 3.1 ขั้นตอนการดำเนินโครงงาน

## 3.1 ศึกษาอัลกอริทึมการเข้ารหัสแบบฮัฟฟ์แมนและขั้นตอนการออกแบบ

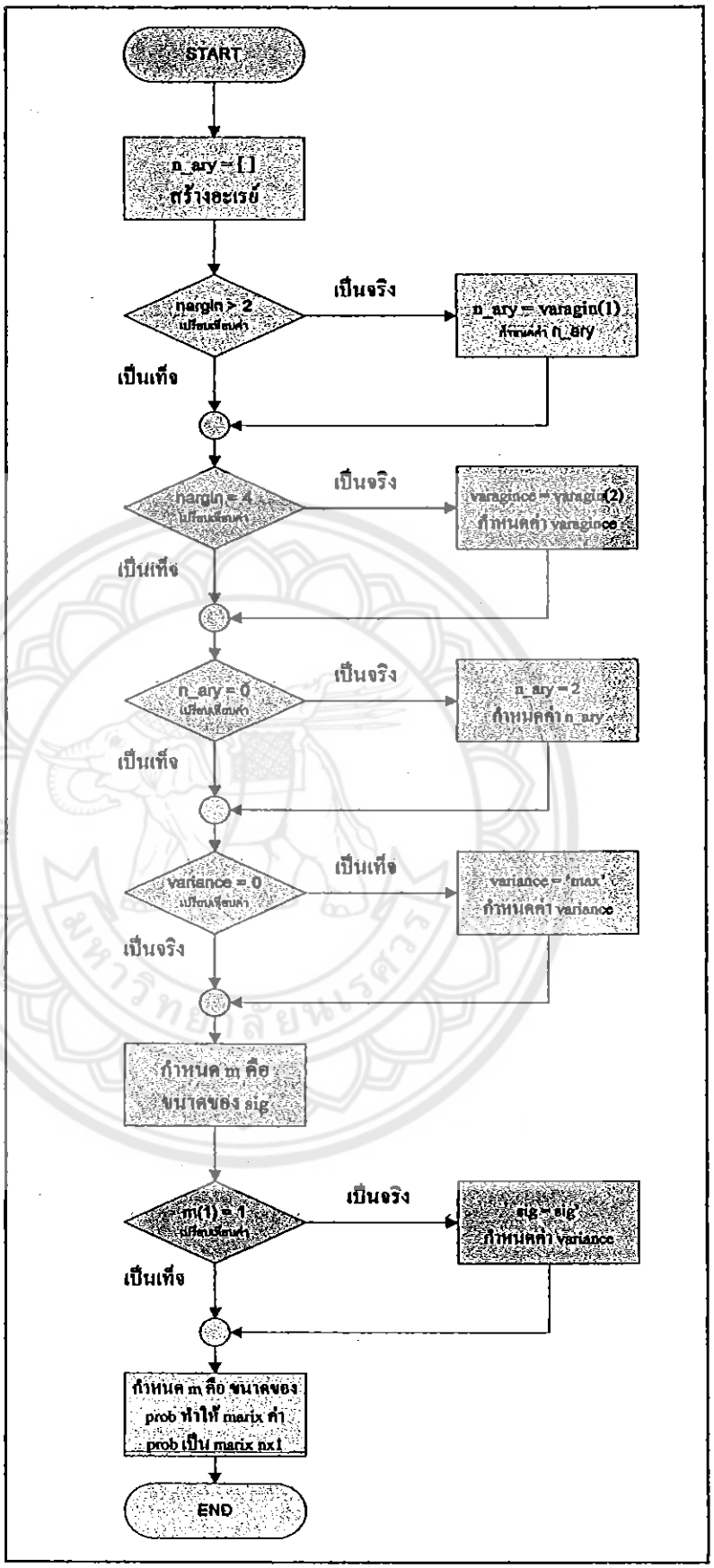
จากบทที่ 2 ศึกษาทฤษฎีและอัลกอลิทึมที่เกี่ยวข้อง เราสามารถนำทฤษฎีและอัลกอลิทึมที่ได้มาออกแบบโปรแกรมการบีบอัดข้อมูลภาพด้วยวิธีการฮัฟฟ์แมนดังแสดงในรูปที่ 3.2 และนำมาเขียนโปรแกรมตารางรหัสฮัฟฟ์แมน และ เขียนโปรแกรมการเข้ารหัสและถอดรหัสฮัฟฟ์แมนได้ซึ่งแสดงในหัวข้อต่อไป




**รูปที่ 3.2** Flowchart โปรแกรมการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมน

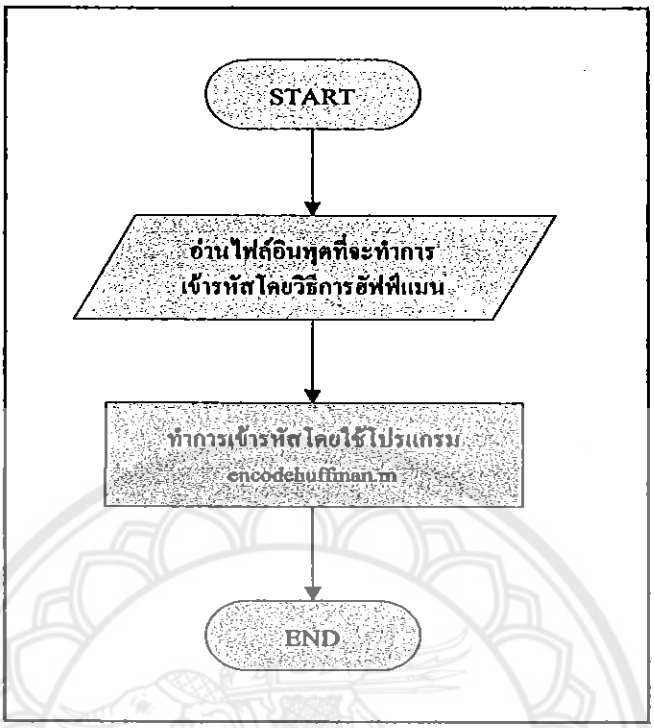
## 3.2 โปรแกรมการสร้างตารางรหัสฮัฟฟ์แมน

การสร้างตารางรหัสฮัฟฟ์แมนแสดงรูปที่ 3.3 และโปรแกรมการสร้างตารางรหัสฮัฟฟ์แมนดูจากภาคผนวก จะใช้ค่าที่ได้นำมาสร้างตารางรหัสตามวิธีของการเข้ารหัสฮัฟฟ์แมน โดยจะเริ่มจากการสร้างแผนภูมิฮัฟฟ์แมนก่อน ทำให้ได้โครงสร้างของแผนภูมิฮัฟฟ์แมน โดยเก็บไว้ในตัวแปรอาเรย์ จากนั้นจะทำการแปลงแผนภูมิฮัฟฟ์แมนให้เป็นรหัสฮัฟฟ์แมนโดยจะทำการเก็บไว้ในตัวแปรอาเรย์เพื่อใช้ในการเข้ารหัสฮัฟฟ์แมนต่อไป




**รูปที่ 3.3** Flowchart ฟังก์ชันสร้างตารางรหัสฮัฟฟ์แมน

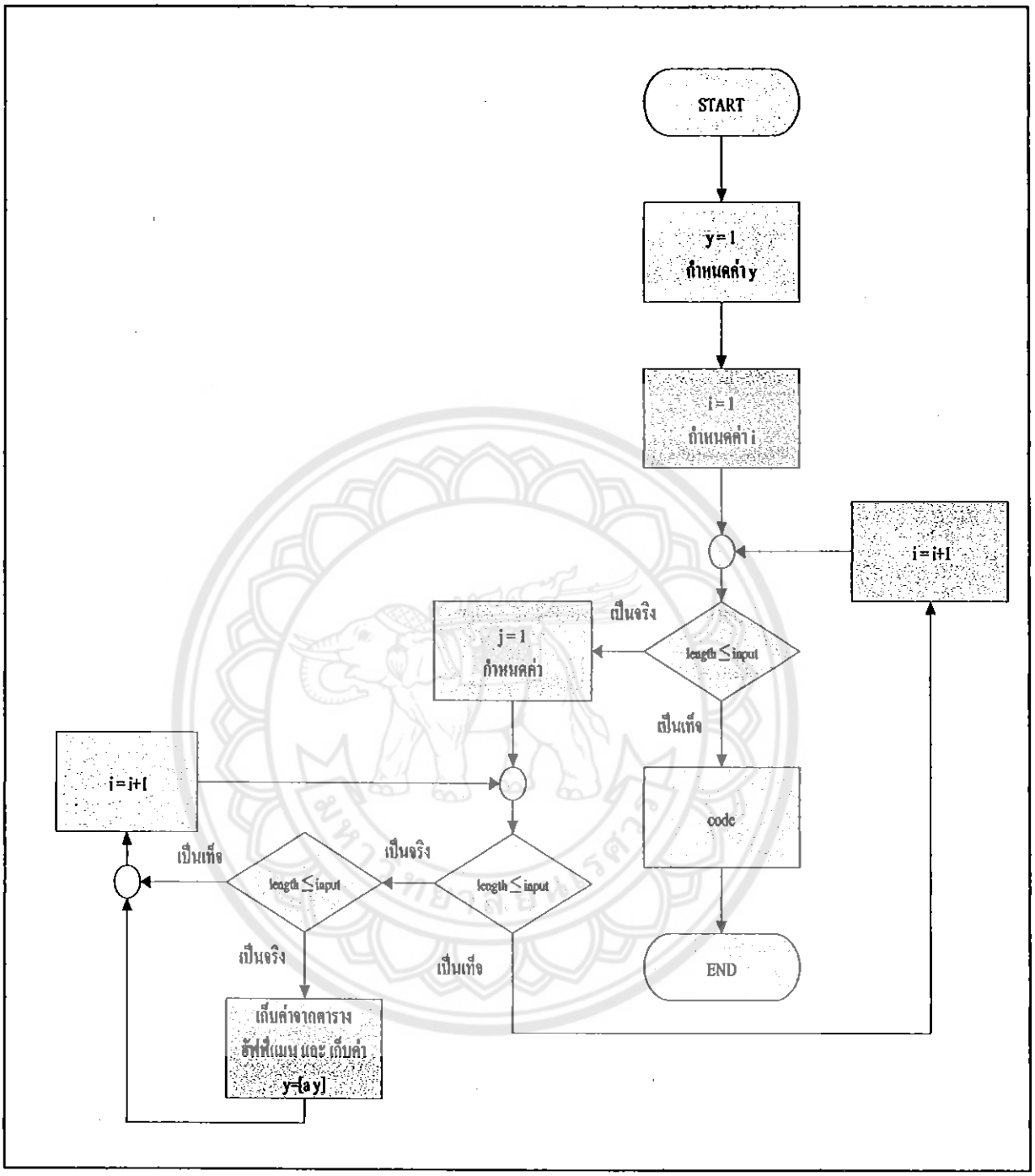
## 3.3 โปรแกรมเข้ารหัสฮัฟฟ์แมน (Huffman Encode)




**รูปที่ 3.4** แผนภาพโปรแกรมการเข้ารหัสฮัฟฟ์แมน (Huffman Encode)

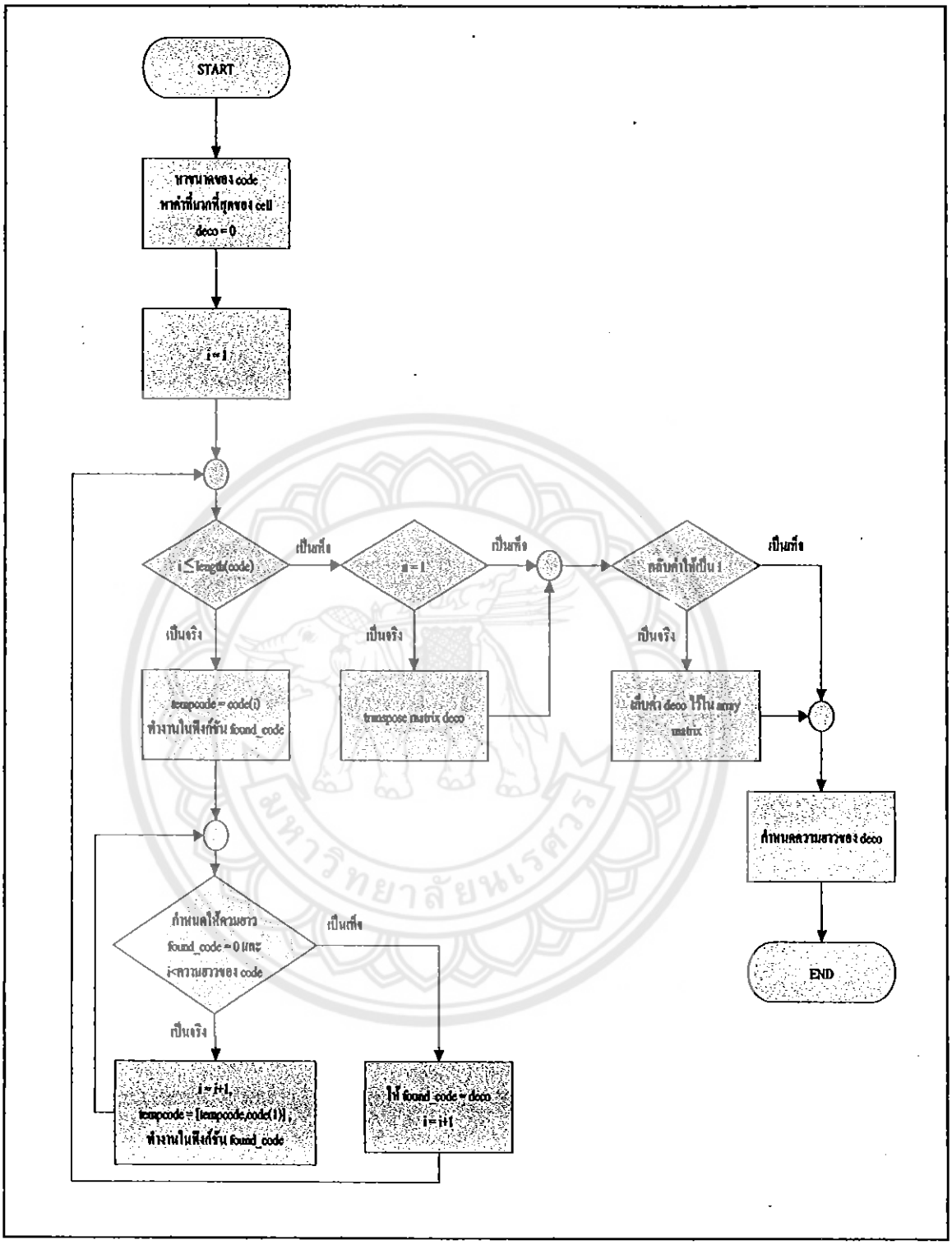
### 3.3.1 ฟังก์ชันการเข้ารหัสฮัฟฟ์แมน

**ฟังก์ชัน** code = encodehuffman(input,dict) **ทำหน้าที่ในการเขียนข้อมูลที่ละบิตโดยจะรวมให้ครบ 8 บิตก่อนแล้วจึงทำการเขียน เนื่องจากรหัสฮัฟฟ์แมนมีความยาวของบิตที่แปรแปลี่ยนได้หรือมีความยาวของจำนวนบิตที่ไม่เท่ากัน ขั้นตอนการทำงานของฟังก์ชันการเข้ารหัสฮัฟฟ์แมนแสดงดังรูปที่ 3.5 และโปรแกรมการเข้ารหัสฮัฟฟ์แมนดูจากภาคผนวก**

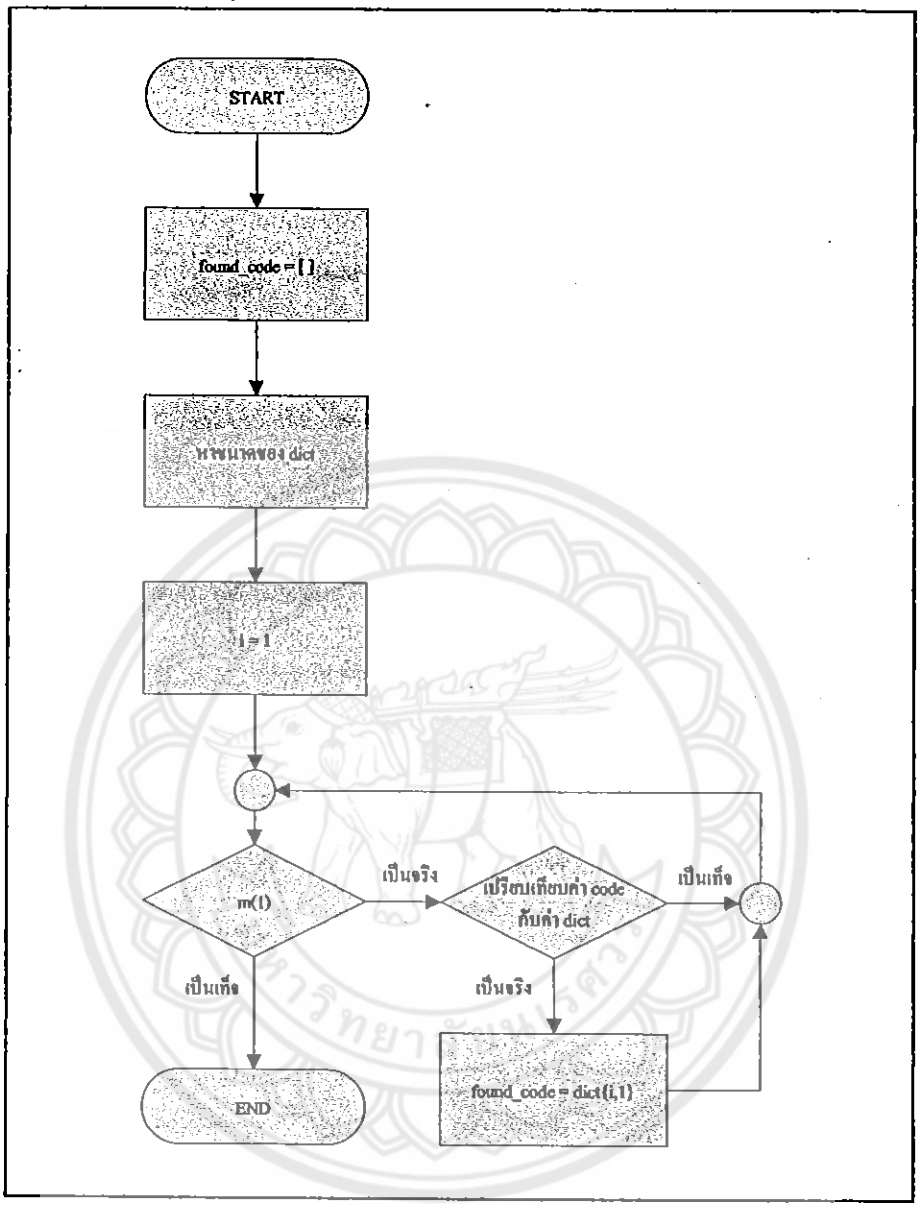



**รูปที่ 3.5** Flowchart ฟังก์ชันการเข้ารหัสแบบฮัฟฟ์แมน

MISSING




**รูปที่ 3.7** Fourchart ฟังก์ชันการถอดรหัสฮัฟฟ์แมน




**รูปที่ 3.7** Flowchart ฟังก์ชันการถอดรหัสฮัฟฟ์แมน (ต่อ)

# บทที่ 4

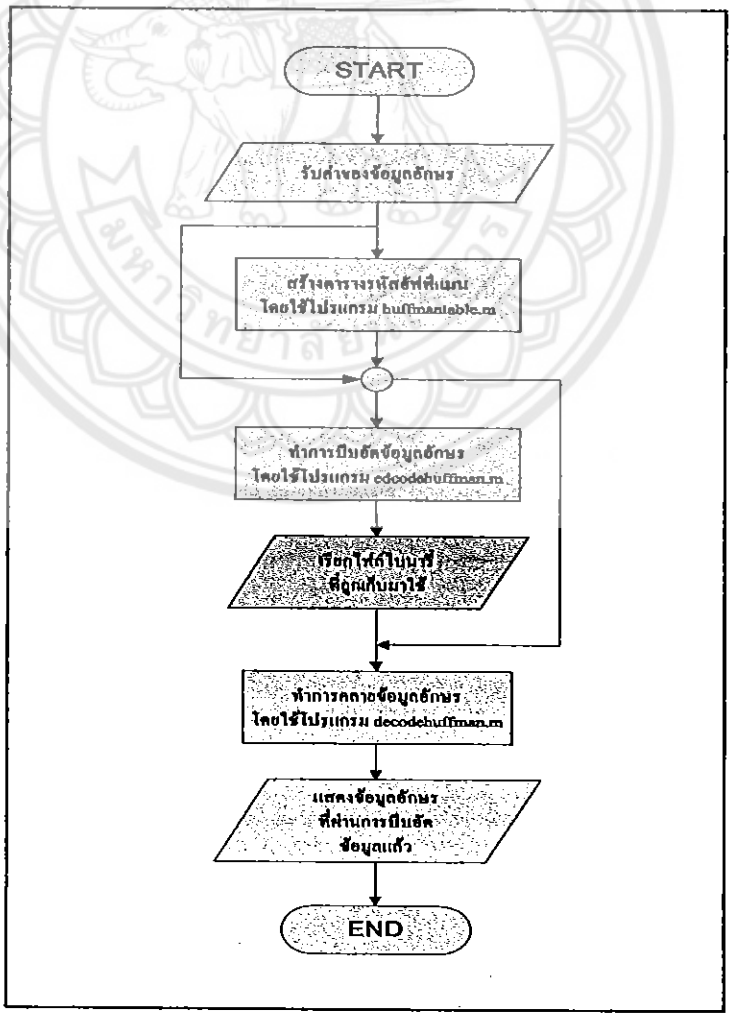
# การทดลอง

## 4.1 วัตถุประสงค์การทดลอง

1. ทดลองการบีบอัดข้อมูลภาพโดยวิธีการแบบฮัฟฟ์แมน เพื่อลดจำนวนบิตของข้อมูลภาพ
2. ทดลองการบีบอัดข้อมูลอักษรโดยวิธีการแบบฮัฟฟ์แมน เพื่อลดจำนวนบิตของข้อมูลอักษร

## 4.2 การเรียกใช้โปรแกรม

โปรแกรมการบีบอัดข้อมูลภาพโดยวิธีการแบบฮัฟฟ์แมนเรียกใช้โปรแกรม hufftest.m มีขั้นตอนการทำงานดังรูปที่ 3.2 (สามารถดูโปรแกรมได้จากภาคผนวก) และการเรียกใช้โปรแกรมบีบอัดข้อมูลอักษร โดยวิธีการแบบฮัฟฟ์แมนจะเรียกใช้โปรแกรม hufftest1.m ดังแสดงในรูปที่ 4.1 (สามารถดูโปรแกรมได้จากภาคผนวก)



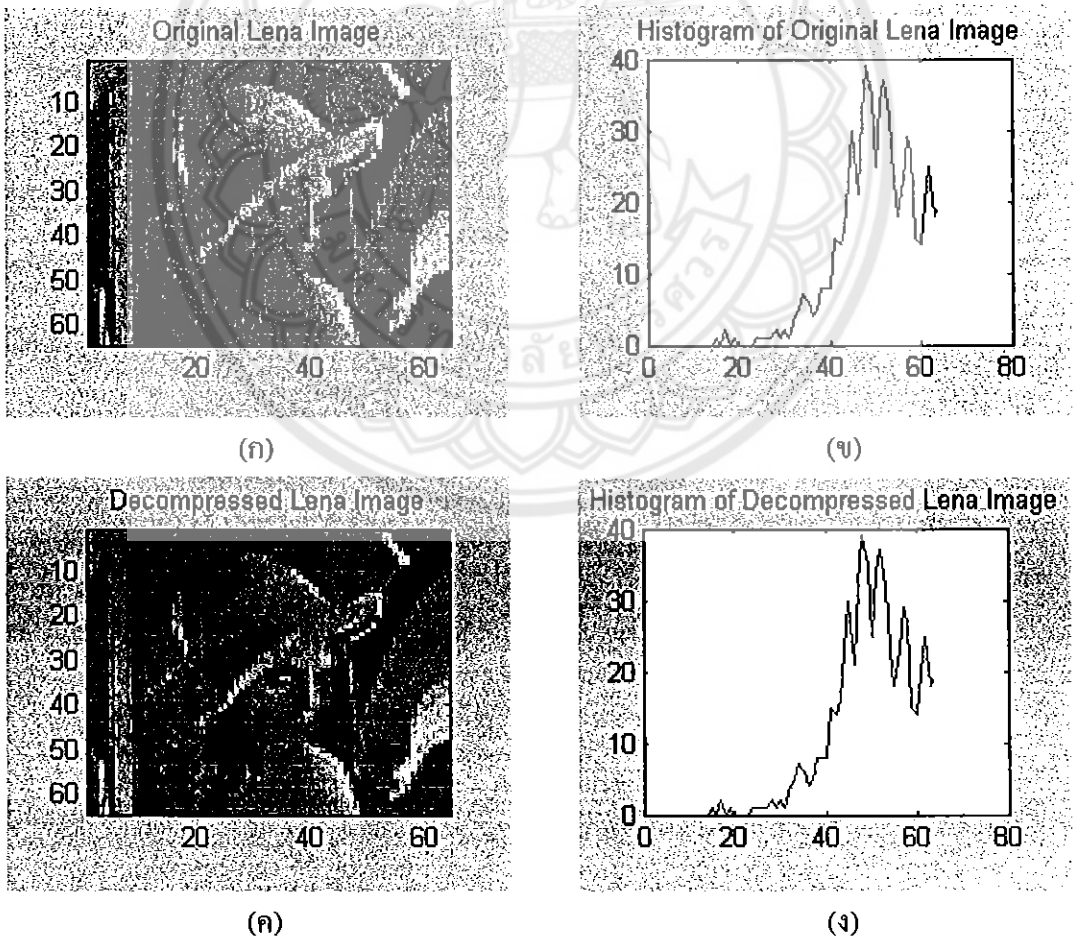

**รูปที่ 4.1** Flowchart โปรแกรมการบีบอัดข้อมูลอักษรโดยวิธีการแบบฮัฟฟ์แมน

## 4.3 การทดลองเข้ารหัสและการถอดรหัสฮัฟฟ์แมน

การทดลองจะทำการเข้ารหัสและถอดรหัสฮัฟฟ์แมนโดยใช้โปรแกรม MATLAB เปรียบเทียบกับการเข้ารหัสฮัฟฟ์แมนโดยจะวัดจากอัตราการบีบอัดข้อมูลภาพ โดยไฟล์ตัวอย่างที่ใช้ในการทดลองเป็นไฟล์ที่เป็นไฟล์ภาพ และไฟล์ข้อมูล ดังนี้

### 4.3.1 ผลการทดลองการบีบอัดข้อมูลภาพ

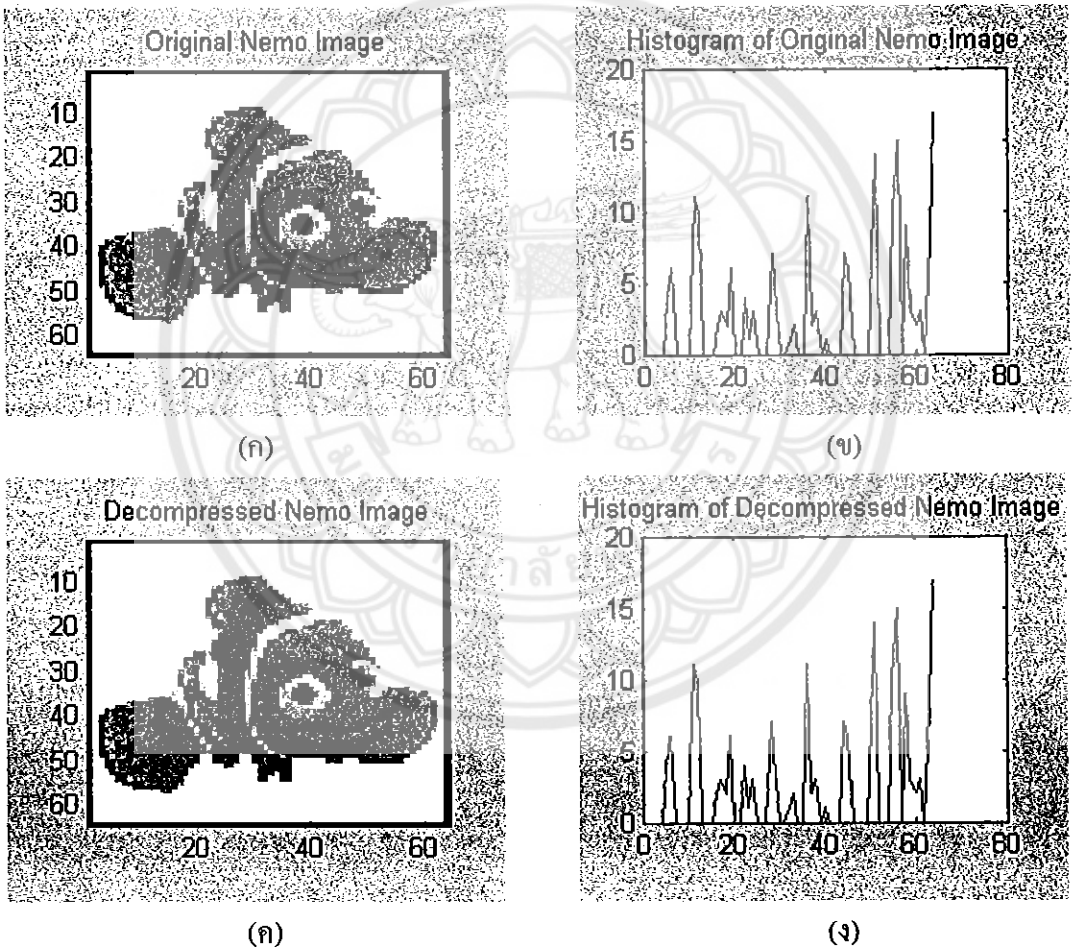
รูปที่ 4.2 (ก) แสดงภาพ Lena ก่อนการบีบอัดข้อมูลภาพซึ่งมีขนาดไฟล์ที่จัดเก็บ 5,174 bytes และรูปที่ 4.2 (ข) แสดงคุณลักษณะกราฟฮีสโตแกรมของภาพต่อจากนั้นเข้าการบีบอัดข้อมูลโดยใช้การเข้ารหัสฮัฟฟ์แมน ได้ไฟล์การบีบอัดข้อมูลภาพซึ่งมีขนาดจัดเก็บ 4,118 bytes และได้คำนวณหาค่าอัตราส่วนการบีบอัดข้อมูลภาพเท่ากับ 1.26 เมื่อได้คลายการบีบอัดข้อมูลภาพจะได้ผลลัพธ์ของภาพแสดงไว้ในรูปที่ 4.2 (ค) ซึ่งได้แสดงผลกราฟฮีสโตแกรมไว้ในรูปที่ 4.2 (ง) ไว้ด้วย ทำการเปรียบเทียบภาพทั้งสองได้ค่า SNR และ PSNR เข้าสู่เลขอนันต์ นั้นแสดงให้เห็นว่าการบีบอัดและการคลายการบีบอัดข้อมูลภาพ Lena มีคุณลักษณะเหมือนกันทุกประการ




**รูปที่ 4.2** แสดงภาพ Lena ก่อนการบีบอัดและหลังการบีบอัดข้อมูลภาพด้วยฮัฟฟ์แมน

( CR = 1.26, SNR = ∞, PSNR = ∞ )

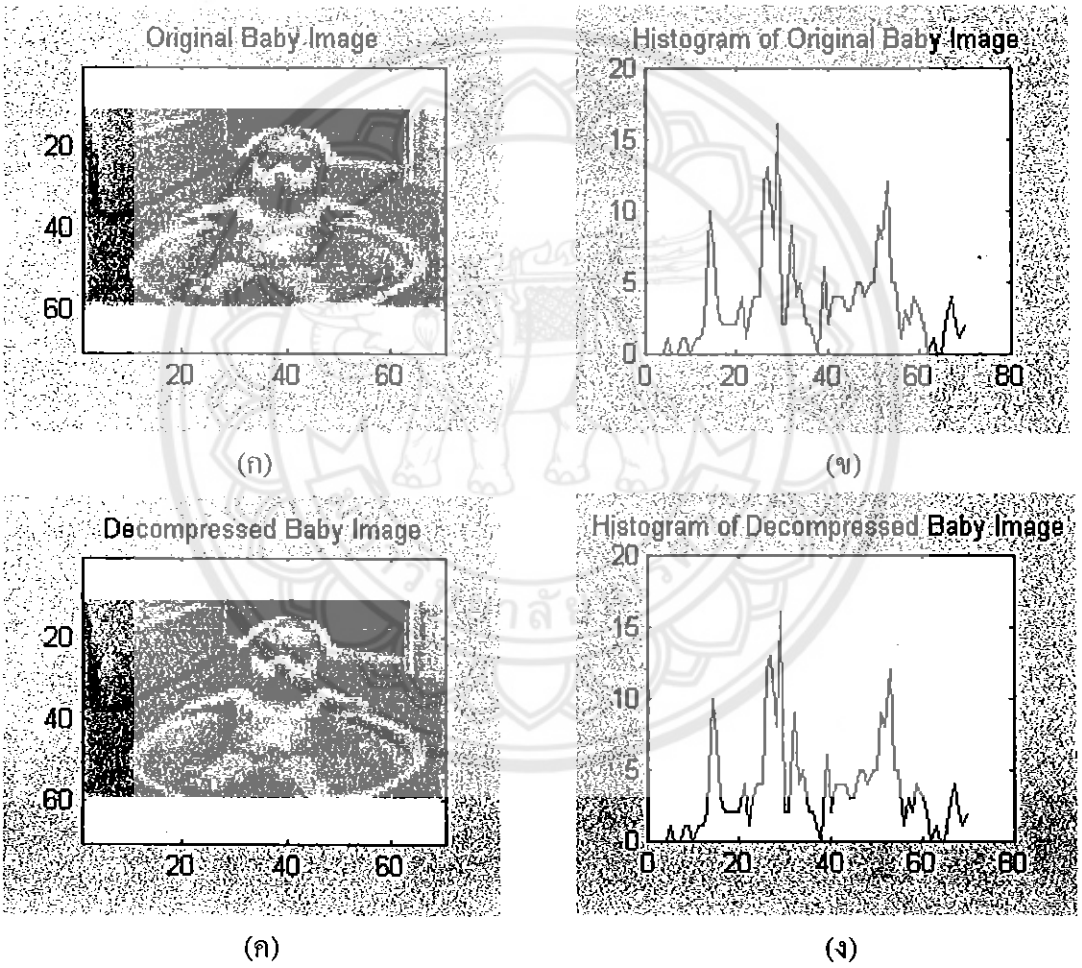
รูปที่ 4.3 (ก) แสดงภาพ Nemo ก่อนการบีบอัดข้อมูลภาพซึ่งมีขนาดไฟล์ที่จัดเก็บ 5,176 bytes และรูปที่ 4.3 (ข) แสดงคุณลักษณะกราฟฮีสโตแกรมของภาพต่อจากนั้นเข้าการบีบอัดข้อมูลโดยใช้การเข้ารหัสฮัฟฟ์แมน ได้ไฟล์การบีบอัดข้อมูลภาพซึ่งมีขนาดจัดเก็บ 3,913 bytes และได้คำนวณหาค่าอัตราส่วนการบีบอัดข้อมูลภาพเท่ากับ 1.32 เมื่อได้คลายการบีบอัดข้อมูลภาพจะได้ผลลัพธ์ของภาพแสดงไว้ในรูปที่ 4.3 (ค) ซึ่งได้แสดงผลกราฟฮีสโตแกรมไว้ในรูปที่ 4.3 (ง) ไว้ด้วย ทำการเปรียบเทียบภาพทั้งสองได้ค่า SNR และ PSNR เข้าสู่เลขอนันต์ นั้นแสดงให้เห็นว่าการบีบอัดและการคลายการบีบอัดข้อมูลภาพ Nemo มีคุณลักษณะเหมือนกันทุกประการ




**รูปที่ 4.3** แสดงภาพ Nemo ก่อนการบีบอัดและหลังการบีบอัดข้อมูลภาพด้วยฮัฟฟ์แมน

( CR = 1.32, SNR = ∞, PSNR = ∞ )

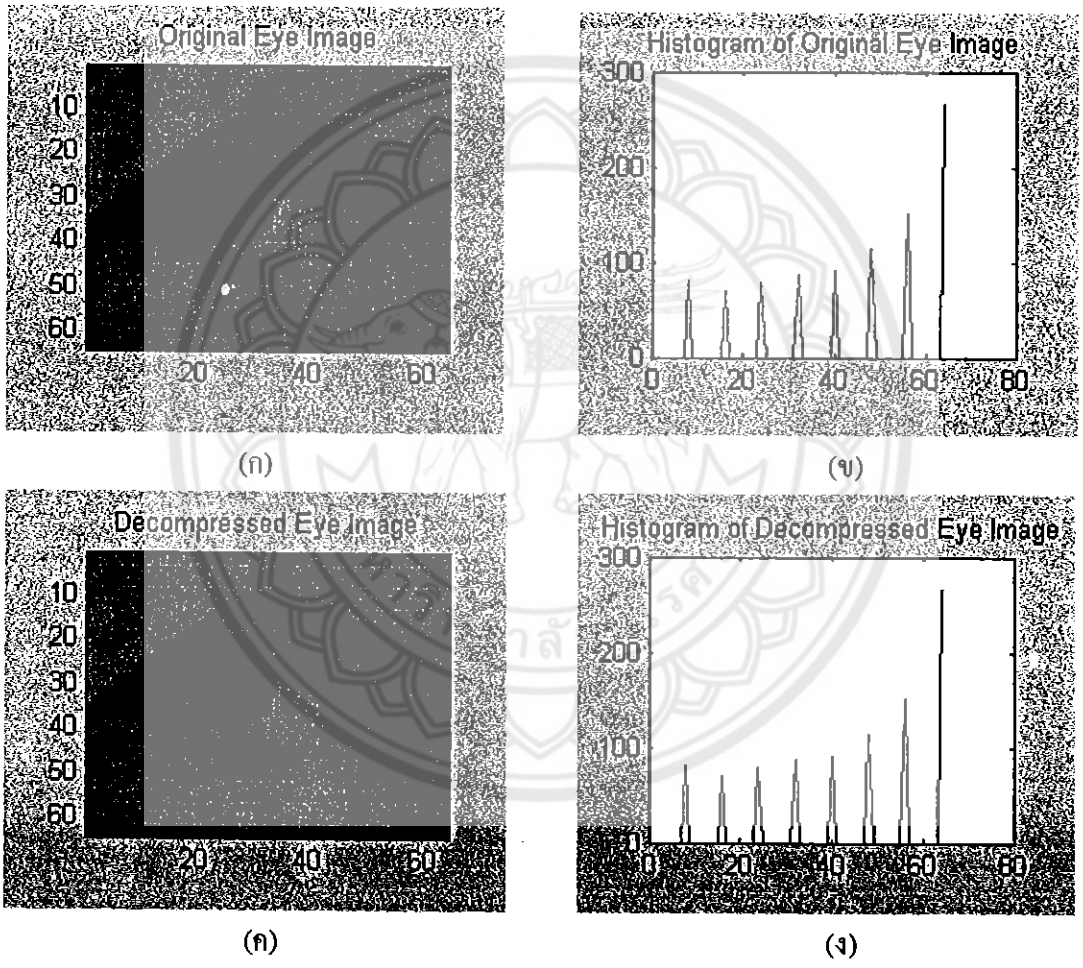
รูปที่ 4.4 (ก) แสดงภาพ Baby ก่อนการบีบอัดข้อมูลภาพซึ่งมีขนาดไฟล์ที่จัดเก็บ 6,118 bytes และรูปที่ 4.4 (ข) แสดงคุณลักษณะกราฟฮีสโตแกรมของภาพต่อจากนั้นเข้าการบีบอัดข้อมูลโดยใช้การเข้ารหัสฮัฟฟ์แมน ได้ไฟล์การบีบอัดข้อมูลภาพซึ่งมีขนาดจัดเก็บ 4,728 bytes และได้คำนวณหาค่าอัตราส่วนการบีบอัดข้อมูลภาพเท่ากับ 1.30 เมื่อได้คลายการบีบอัดข้อมูลภาพจะได้ผลลัพธ์ของภาพแสดงไว้ในรูปที่ 4.4 (ค) ซึ่งได้แสดงผลกราฟฮีสโตแกรมไว้ในรูปที่ 4.4 (ง) ไว้ด้วย ทำการเปรียบเทียบภาพทั้งสองได้ค่า SNR และ PSNR เข้าสู่เลขอนันต์ นั้นแสดงให้เห็นว่าการบีบอัดและการคลายการบีบอัดข้อมูลภาพ Baby มีคุณลักษณะเหมือนกันทุกประการ




**รูปที่ 4.4** แสดงภาพ Baby ก่อนการบีบอัดและหลังการบีบอัดข้อมูลภาพด้วยฮัฟฟ์แมน

( CR = 1.30, SNR = ∞, PSNR = ∞ )

รูปที่ 4.5 (ก) แสดงภาพ Eye ก่อนการบีบอัดข้อมูลภาพซึ่งมีขนาดไฟล์ที่จัดเก็บ 5,172 bytes และรูปที่ 4.5 (ข) แสดงคุณลักษณะกราฟฮีสโตแกรมของภาพต่อจากนั้นเข้าการบีบอัดข้อมูลโดยใช้การเข้ารหัสฮัฟฟ์แมน ได้ไฟล์การบีบอัดข้อมูลภาพซึ่งมีขนาดจัดเก็บ 4,004 bytes และได้คำนวณหาค่าอัตราส่วนการบีบอัดข้อมูลภาพเท่ากับ 1.30 เมื่อได้คลายการบีบอัดข้อมูลภาพจะได้ผลลัพธ์ของภาพแสดงไว้ในรูปที่ 4.5 (ค) ซึ่งได้แสดงผลกราฟฮีสโตแกรมไว้ในรูปที่ 4.5 (ง) ไว้ด้วย ทำการเปรียบเทียบภาพทั้งสองได้ค่า SNR และ PSNR เข้าสู่เลขอนันต์ นั้นแสดงให้เห็นว่าการบีบอัดและการคลายการบีบอัดข้อมูลภาพ Eye มีคุณลักษณะเหมือนกันทุกประการ




**รูปที่ 4.5** แสดงภาพ Eye ก่อนการบีบอัดและหลังการบีบอัดข้อมูลภาพด้วยฮัฟฟ์แมน

( CR = 1.30, SNR = ∞, PSNR = ∞ )

จากผลการทดลองการเข้ารหัสและถอดรหัสไฟล์ภาพแบบฮัฟฟ์แมน ดังกล่าวข้างต้นสามารถสรุปผลที่ได้จากการบีบอัดข้อมูลภาพโดยวิธีฮัฟฟ์แมนดังตารางที่ 4.1

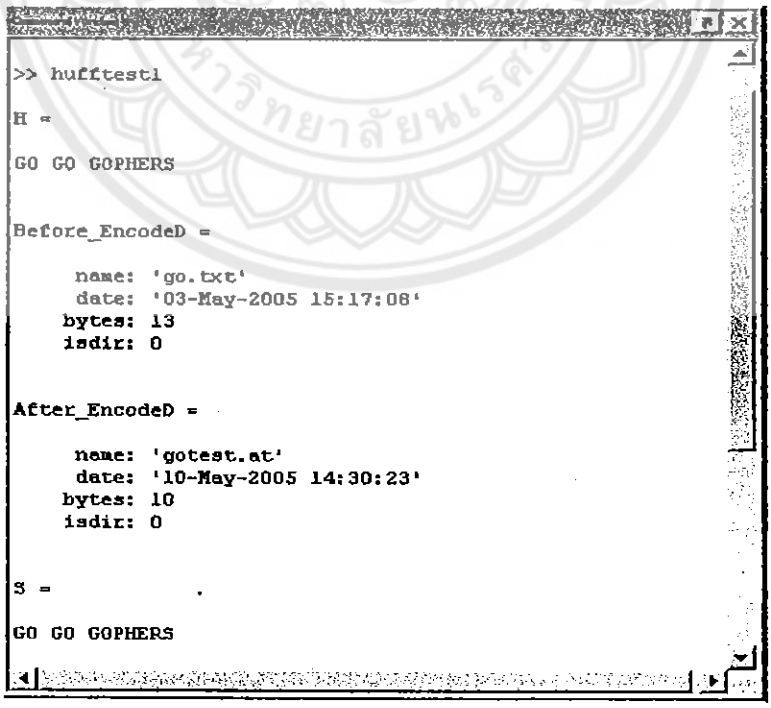


**ตารางที่ 4.1 สรุปผลที่ได้จากการบีบอัดข้อมูลภาพโดยวิธีการแบบฮัฟฟ์แมนทั้ง 4 กรณี**

| ลำดับที่ | ชื่อภาพ | CR | SNR (dB) | PSNR (dB) |
|---|---|---|---|---|
| 1 | Lena | 1.26 | ∞ | ∞ |
| 2 | Nemo | 1.32 | ∞ | ∞ |
| 3 | Baby | 1.30 | ∞ | ∞ |
| 4 | Eye | 1.30 | ∞ | ∞ |

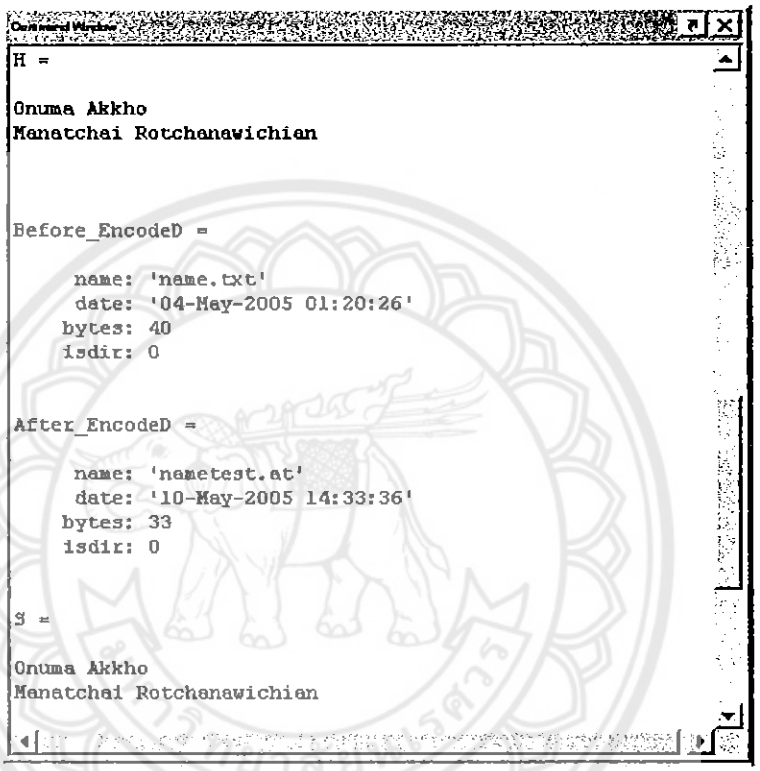
### 4.3.2 ผลการทดลองการบีบอัดข้อมูลอักษร

รูปที่ 4.6 แสดงผลการทดลองไฟล์ go.txt โดยที่ค่า H คือ ข้อมูลที่ใช้ในการทดลองก่อนเข้ารหัสแบบฮัฟฟ์แมน กำหนดให้ Before_EncodeD คือ ค่าของขนาดไฟล์ go.txt ซึ่งมีขนาดไฟล์ที่จัดเก็บ 13 bytes ค่า After_EncoceD คือ ค่าของขนาดไฟล์ข้อมูลที่ถูกบีบอัดข้อมูลแล้วจะเก็บในไฟล์ gotest.at ขนาดไฟล์ที่จัดเก็บ 10 bytes และ S คือ ข้อมูลที่ผ่านการคลายการบีบอัดข้อมูลอักษรโดยวิธีการแบบฮัฟฟ์แมน




**รูปที่ 4.6 แสดงผลการทดลองไฟล์ go.txt**

รูปที่ 4.7 แสดงผลการทดลองไฟล์ name.txt โดยที่ค่า H คือ ข้อมูลที่ใช้ในการทดลองก่อนเข้ารหัสแบบฮัฟฟ์แมน กำหนดให้ Before_EncodeD คือ ค่าของขนาดไฟล์ name.txt ซึ่งมีขนาดไฟล์ที่จัดเก็บ 40 bytes ค่า After_EncoceD คือ ค่าของขนาดไฟล์ข้อมูลที่ถูกบีบอัดข้อมูลแล้วจะเก็บในไฟล์ nametest.at ขนาดไฟล์ที่จัดเก็บ 33 bytes และ S คือ ข้อมูลที่ผ่านการคลายการบีบอัดข้อมูลอักษรโดยวิธีการแบบฮัฟฟ์แมน




รูปที่ 4.7 ผลแสดงการทดลองไฟล์ name.txt

จากผลการทดลองการเข้ารหัสและถอดรหัสข้อมูลอักษรแบบฮัฟฟ์แมน ได้สรุปผลที่ได้จากการบีบอัดข้อมูลโดยวิธีฮัฟฟ์แมนดังตารางที่ 4.2

**ตารางที่ 4.2** สรุปผลที่ได้จากการบีบอัดข้อมูลอักษรโดยวิธีการแบบฮัฟฟ์แมนทั้ง 2 กรณี

| ชื่อไฟล์ | CR | SNR (dB) | PSNR (dB) |
|---|---|---|---|
| "Go.txt" | 1.30 | ∞ | ∞ |
| "Name.txt" | 1.21 | ∞ | ∞ |

## 4.4 ผลการวิเคราะห์การทดลอง

จากผลการทดลองค่าในตารางที่ 4.1 สรุปผลที่ได้จากการบีบอัดข้อมูลภาพโดยวิธีการแบบฮัฟฟ์แมนทั้ง 4 กรณี ค่าของอัตราการบีบอัดข้อมูลอยู่ที่ประมาณ 1.26 – 1.32 ในขณะที่ค่า SNR และ PSNR มีค่าเข้าสู่เลขอนันต์ทุกกรณีนั้น แสดงให้เห็นว่าการบีบอัดและการคลายการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนมีคุณลักษณะเหมือนกันทุกประการ

จากผลการทดลองค่าในตารางที่ 4.2 สรุปผลที่ได้จากการบีบอัดข้อมูลอักษรโดยวิธีการแบบฮัฟฟ์แมนทั้ง 2 กรณี ค่าของอัตราการบีบอัดข้อมูลอักษรอยู่ที่ประมาณ 1.21 – 1.30 ในขณะที่ค่า SNR และ PSNR มีค่าเข้าสู่เลขอนันต์ทุกกรณีนั้นแสดงให้เห็นว่าการบีบอัดและการคลายการบีบอัดข้อมูลอักษรแบบฮัฟฟ์แมน มีคุณลักษระเหมือนกันทุกประการ

ดังนั้นการบีบอัดข้อมูลภาพและข้อมูลอักษรโดยวิธีการฮัฟฟ์แมน จะมีการบีบอัดข้อมูลได้ค่า CR สูงสุด 1.32 และคุณภาพของข้อมูลที่ได้หลังการบีบอัดแบบฮัฟฟ์แมนก็เหมือนข้อมูลต้นแบบทุกประการ

# บทที่ 5
# บทสรุป

## 5.1 สรุปผลการทดลอง

จากบทที่ 4 ผลการทดลองที่ได้ทำให้ทราบว่า การบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนสามารถบีบอัดข้อมูลภาพได้ค่า CR สูงสุด 1.32 และการบีบอัดข้อมูลอักษรแบบฮัฟฟ์แมนสามารถบีบอัดข้อมูลอักษรได้ค่า CR สูงสุด 1.30 และไม่มีการสูญเสียข้อมูลภาพและข้อมูลอักษรหลังการคลายการบีบอัด ซึ่งค่า SNR และ PSNR มีค่าเข้าสู่เลขอนันต์ทั้งสองกรณี

## 5.2 ปัญหาในการทดลองและแนวทางแก้ไข

**ปัญหาที่เกิดขึ้น**

ถ้าใช้ไฟล์ภาพขนาดใหญ่จะใช้เวลาในการรันโปรแกรมเวลานาน

**แนวทางการแก้ไขปัญหา**

เนื่องจากการเข้ารหัสและการถอดแบบฮัฟฟ์แมนนั้นจะทำทีละตัว ทำให้ใช้เวลานานในการรันโปรแกรม ดังนั้นจึงมีการกำหนดขนาดภาพให้เล็กลงเพื่อจะใช้เวลาในการรันโปรแกรมน้อยลง โดยกำหนดให้มีขนาดอยู่ที่ประมาณ 64x64 pixels

## 5.3 แนวทางการพัฒนาในอนาคต

เนื่องจากการพยายามบีบอัดข้อมูลให้มีขนาดเล็กลงหลังการเข้ารหัสฮัฟฟ์แมน แต่ในการรันโปรแกรมที่มีไฟล์ข้อมูลขนาดใหญ่ยังใช้เวลาในการรันโปรแกรมนาน ดังนั้นควรที่จะเขียนโปรแกรมที่สามารถใช้เวลาในการรันโปรแกรมน้อยแต่สามารถบีบอัดข้อมูลที่มีไฟล์ขนาดใหญ่ได้

# เอกสารอ้างอิง


[1] รศ.ดร.มนัส สังวรศิลป์ . **คู่มือการใช้งาน MATLAB ฉบับสมบูรณ์ .**กรุงเทพมหานคร : อินโฟเพรส, 2543

[2] David A. Huffman. **" A Method for the Construction of Minimum-Redundancy Code ",** roc. IRE., Vol.40,pp.1098-1101, Sept. 1952.

[3] Ming-I Lu, and Chang-Fuu Chean. **" An Encoding Procedure and a Decoding Procedure for a New Modifier Huffman Code ",** IEEE trans. On Acoustics Speech and Signal Processing, Vol. 38, No.1, pp. 128-130, Jan. 1990

# ภาคผนวก

## โปรแกรมการบีบอัดข้อมูลภาพโดยวิธีการฮัฟฟ์แมน

## SOURCE PROGRAM

1. โปรแกรมเรียกใช้การบีบอัดข้อมูลภาพ

```
%==========================================================
% (Test Huffman code)
% File name : hufftest.m
% By :  Mr. Manutchai  Rojanavachian    ID 44370336
%       Miss. Onuma    Akkho            ID 44370567
%==========================================================

clear all
close all

input = double(imread('lena.bmp','bmp'));          % Data to encode
sig = input(:);

histo=medhisto(input);                             % Histogram is input

[M N]=size(input);                                 % size is input
actualsig = sig';
sym = sort(actualsig);
symbols = 0:1:sym(length(sym));
p = symbols/sum(symbols);                          % Probability distribution
dict = huffmantable(symbols,p);                    % Create dictionary.


% Encode of Huffman code
code = encodehuffman(actualsig,dict);

% Save Files
fid = fopen('test.kj','wb');                       % open file in binary write mode
fwrite(fid,code,'ubit1');                          % write data to file
fclose(fid);                                       % close file
[a b] = size(code);
```

```
%read in the same file
fid  = fopen('test.kj','rb');                    % open file
data = fread(fid,[a b],'ubit1');                 % read in the data
fclose(fid);                                     % close file

% Show size files
Before_EncodeD=dir('lena.bmp')
After_EncodeD=dir('test.kj')

% Decode of Huffman code
deco = decodehuffman(data, dict);                % Encode the data.
desig =reshape(deco,M,N);

histo1=medhisto(desig);                          % Histogram is desig.
% ------------------------------------------------------------------
% Check Error
ER = (input-desig);
ERR = ER.^2;
nesig = input.^2;
Error = (sum(ERR(:)))/M*N;
SNR = 10*log10((sum(sig(:))/M*N)/Error)
PSNR = 10*log10(max(sig(:))/Error)
% ------------------------------------------------------------------
% Display(Show)
figure(1)
subplot(2,2,1),image(input),colormap(gray(256)),title('Original Lena Image')
subplot(2,2,2),plot(histo),title('Histogram of Original Lena Image')
figure(2)
subplot(2,2,1),image(desig),colormap(gray(256)),title('Decompressed Lena Image')
subplot(2,2,2),plot(histo1),title('Histogram of Decompressed Lena Image')
```

2. โปรแกรมเรียกใช้การบีบอัดข้อมูลไฟล์ .txt

```
% (Test Huffman code)
% File name : hufftest1.m
% By :  Mr. Manutchai  Rojanavachian    ID 44370336
%      Miss. Onuma    Akkho             ID 44370567
% ==========================================================
clear all
close all

fif = fopen('name.txt','r');          % open file in txt.
input = fread(fif);                   % read file in txt.
H = char(input')                      % show to file.

sig = input(:);
[M N]=size(sig);
actualsig = sig';
sym = sort(actualsig);
symbols = 0:1:sym(length(sym));
p = symbols/sum(symbols);             % Probability distribution
dict = huffmantable(symbols,p);       % Create dictionary.
code = encodehuffman(actualsig,dict); % Encode the data.


% Save Files
fid = fopen('nametest.at','wb');      % open file in binary write mode
fwrite(fid,code,'ubit1');             % write data to file
fclose(fid);                          % close file
[a b] = size(code);

%read in the same file
fid  = fopen('nametest.at','rb');     % open file
```

```
data = fread(fid,[a b],'ubit1');          % read in the data
fclose(fid);                              % close file

% Show size files
Before_EncodeD=dir('name.txt')
After_EncodeD=dir('nametest.at')


% Huffman code (decode)
dsig = decodehuffman(data,dict);          % Decode the Huffman code.
S = char(dsig)

% Save Files
fid = fopen('name1.txt','w');             % open file in write mode
fwrite(fid,dsig,'ubit8');                 % write data to file
fclose(fid);                              % close file

% Check Error
ER = (H-dsig);
ERR = ER.^2;
nesig = input.^2;
Error = (sum(ERR(:)))/M*N;
SNR = 10*log10((sum(sig(:))/M*N)/Error)
PSNR = 10*log10(max(sig(:))/Error)
```

3. โปรแกรมสร้างตารางฮัฟฟ์แมน

```
% ==========================================================
% FUNCTION dict = huffmantable(sig,prob);
% This function is Table of huffman code
% sig is input
% prob is Probability distribution
% dict is Table of huffman code
% ==========================================================
function dict = huffmantable(sig,prob);

n_ary = [];
variance = '';
msg=nargchk(2,4, nargin);
if nargin > 2
        n_ary = varargin(1);
end
if nargin == 4
        variance = varargin(2);
end
if isempty(n_ary)
   n_ary = 2;                          % default value is binary encryption
end
if ( variance )                        % if variance contains a non-null string do nothing
else
   variance = 'max';                   % default is maximum variance Huffman code
end
% Make sure that internally all vectors are represented as column vectors
m = size(sig);
if( m(1) == 1 )
   sig = sig';
end
m = size(prob);
```

```
prob = prob(:);
% Make sure that the input symbols are in a cell array format
if ~iscell(sig)
        [m,n] = size(sig);
        sig = mat2cell(sig, ones(1,m), ones(1,n)) ;
end

% === Input parameters are all set ===
% === Create the dictionary ===
% Create tree nodes with the signals and the corresponding probabilities
huff_tree = struct('signal', [], 'probability', [],...
   'child', [], 'code', [], 'origOrder', -1);
for i=1:length( sig )
   huff_tree(i).signal = sig{i};
   huff_tree(i).probability = prob(i);
        huff_tree(i).origOrder = i;
end
% Sort the signal and probability vectors based on ascending order of
% probability
[s, i] = sort(prob);
huff_tree = huff_tree(i);
huff_tree = create_tree(huff_tree, n_ary, variance);          % create a Huffman tree
[huff_tree,dict,avglen] = create_dict(huff_tree,{},0, n_ary);          % create the codebook

% The next few lines of code are to sort the dictionary.
% If sorting based on original order then use dict{:,4}.
% If sorting based on the lenght of code, then use  dict{:,3}.
[dictsort,dictsortorder] = sort([dict{:,4}]);
finaldict = {};
for i=1:length(dictsortorder)
   finaldict{i,1} = dict{dictsortorder(i), 1};
   finaldict{i,2} = dict{dictsortorder(i), 2};
```

```
end
dict = finaldict;
%------------------------------------------------------------------------
% Function: huff_tree
% Input: An array of structures to be arranged into a Huffman tree
% Utility: This is a recursive algorithm to create the Huffman Code
%          tree. This is a recursive function

function huff_tree = create_tree(huff_tree, n_ary, variance)

% if the length of huff_tree is 1, it implies there is no more than one
% node in the array of nodes. This is the termination condition for the
% recursive loop
if( length(huff_tree) <= 1)
   return;
end

% Combine the first n_ary (lowest probability) number of nodes under one
% parent node, remove these n_ary nodes from the list of nodes and add
% the new parent node that was just created
temp = struct('signal', [], 'probability', 0, ...
   'child', [], 'code', []);
for i=1:n_ary
   if( length(huff_tree) == 0), break; end
   temp.probability = temp.probability + huff_tree(1).probability;          % for ascending
order
   temp.child{i} = huff_tree(1);
         temp.origOrder = -1;
   huff_tree(1) = [];
end
if( strcmpi(variance, 'min') == 1 )
   huff_tree = insertMinVar(huff_tree, temp);
```

```
else
   huff_tree = insertMaxVar(huff_tree, temp);
end
% create a Huffman tree from the reduced number of free nodes
huff_tree = create_tree(huff_tree, n_ary, variance);
return;

%-------------------------------------------------------------------------
% This function will insert a node in the sorted list such that the
% resulting list will be sorted (ascending). If there exists node with the
% same probability as the new node, the new node  is placed after these
% same value nodes.
function huff_tree = insertMaxVar(huff_tree, newNode)
sortedOn = [huff_tree.probability];
i = 1;
while i <= length(huff_tree) & ...
      newNode.probability > huff_tree(i).probability
   i = i+1;
end
huff_tree = [huff_tree(1:i-1) newNode huff_tree(i:end)];

%-------------------------------------------------------------------------
% This function does a pre-order traversal of the tree to create the codes
% for each leaf node. This is a recursive function

function [huff_tree,dict,total_wted_len] = create_dict(huff_tree,dict,total_wted_len, n_ary)

% Check if the current node is a leaf node If it is, then add the signal on
% this node and its corresponding code to the dictionary global n_ary
if( length(huff_tree.child) == 0 )
   dict{end+1,1} = huff_tree.signal;
   dict{end, 2} = huff_tree.code;
```

```
    dict{end, 3} = length(huff_tree.code);
        dict{end, 4} = huff_tree.origOrder;
    total_wted_len = total_wted_len + length(huff_tree.code)*huff_tree.probability;
    return;
end
num_childrens = length(huff_tree.child);
for i = 1:num_childrens
    huff_tree.child{i}.code = [huff_tree(end).code, (num_childrens-i)];
    [huff_tree.child{i}, dict, total_wted_len] = ...
        create_dict(huff_tree.child{i}, dict, total_wted_len, n_ary);
end
```

## 4. โปรแกรมเข้ารหัสแบบฮัฟฟ์แมน

```
% ==========================================================
% FUNCTION : code = encodehuffman(input,dict);
% File name : hufftest.m
% By :  Mr. Manutchai  Rojanavachian    ID 44370336
%       Miss. Onuma    Akkho            ID 44370567
% This function is encodehuffman
% input is input signal
% dict is table's huffman code
% code is output's binary
% ==========================================================
function code = encodehuffman(input,dict);


y = 0 ;
for i = 1:1:length(input),
   for j = 1:1:length(dict),
      if input(i) == dict{j,1},
         a=dict{j,2};
         dict{j,1};
         y = [a y];
      end
   end
end
code = y(1:1:end-1) ;
```

5. โปรแกรมถอดรหัสฮัฟฟ์แมน

```
%=========================================================================
% FUNCTION : deco = decodehuffman(code,dict);
% By :  Mr. Manutchai  Rojanavachian    ID 44370336
%       Miss. Onuma    Akkho            ID 44370567
% This function is decodehuffman
% code is input's binary
% dict is Table's huffman code
% deco is output
%
=========================================================================

function deco = decodehuffman(code, dict)


msg=nargchk(2,2, nargin);
[m,n] = size(code);
isSigNonNumeric = max(cellfun('isclass', {dict{:,1}}, 'char') );
deco = {};
i = 1;
while(i <= length(code))
   tempcode = code(i);
   found_code = is_a_valid_code(tempcode, dict);
   while(length(found_code) == 0 & i < length(code))
      i = i+1;
      tempcode = [tempcode, code(i)];
      found_code = is_a_valid_code(tempcode, dict);
   end
         deco{end+1} = found_code;
   i=i+1;
end
if( n == 1 )        % if input was a column vector
  deco = deco';     % the decoded output should also be a column vector
```

```
end


if ( ~isSigNonNumeric )
        deco = cell2mat(deco);
end
deco = deco(length(deco):-1:1);
%--------------------------------------------------------------------------
function found_code = is_a_valid_code(code, dict)
found_code = [];
m = size(dict);
for i=1:m(1)
  if ( isequal(code, dict{i,2}) )
    found_code = dict{i,1};
    return;
  end
end
```

6. โปรแกรมหาฮีตโตแกรมของข้อมูล

```
% ==========================================================
% HISTO Display histogram of image data.
% L,H]=HISTO(I) displays a histogram with
% H(0),H(1),...,H(L),...,H(255)
% for the intensity image I of L =0,1,2,...,255.
% ==========================================================

function h = medhisto(xx);
h = [];
for gl = 1: 1: length(xx)
   [indx,indy,val] =  find(xx==gl);
   h               =  [h sum(val)];
end
```

# ประวัติผู้เขียนโครงงาน

ชื่อ นายมนัสชัย โรจนวิเชียร

ภูมิลำเนา 214/93 หมู่ที่ 9 ต.นครสวรรค์ตก อ.เมือง จ.นครสวรรค์

ประวัติการศึกษา

- จบการศึกษาชั้นมัธยมศึกษาจากโรงเรียนนครสวรรค์
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรี ชั้นปีที่ 4

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยนเรศวร

E-Mail : **nompang506@hotmail.com**

ชื่อ นางสาวอรอุมา อักโข

ภูมิลำเนา 395/15 หมู่ที่ 11 ต.แสนสุข อ.วารินชำราบ จ.อุบลราชธานี

ประวัติการศึกษา

- จบการศึกษาชั้นมัธยมศึกษาจากโรงเรียนนารีนุกูล
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรี ชั้นปีที่ 4

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยนเรศวร

E-Mail : **onuma_i2000@hotmail.com**